

ATM LANE SIMULATOR

Submitted by

**Ching Kim Joo
WEK 98033**

Under Supervision of

MR. LING TECK CHAW

*Dissertation submitted in partial fulfillment of the requirement for the
Degree of Bachelor of Computer Science*



**Faculty of Computer Science and
Information Technology,
University of Malaya
2000/2001**



ABSTRACT

This project describes the development of ATM Local Area Network Emulation (LANE) simulator components. In LANE, an ATM network is configured to simulate an Ethernet or token ring network and coexist with the legacy network without making any modification to them.

The development of this simulator incorporates several important features. The most important feature of the simulator is that it is build based on object-oriented approach, which is able to provide the benefits of reusability, flexibility and extendibility. The second important feature is that it supports multithreaded operations. This means that every object able to act simultaneously. Besides that, the simulator is web-enabled, which means the simulator can be accessed from the Internet and thus increase the usability of the network simulator.

This network simulator can helps user to understand the operation of LANE and provides the benefits of testing the network without making any investment on the real network. Thus, it can save the cost in terms of unnecessary restructuring for experimentation. Further research on LANE can be carried out with this simulator,

ACKNOWLEDGEMENT

I would like to express my utmost gratitude to Mr. Ling Teck Chaw, supervisor of this project for his patience guidance and advice throughout the whole development of the project. His contribution is truly appreciated. Special thanks to Mr. Ibrahim A. Abdulrrazaq for being a considerate and kind moderator.

Moreover, I would like to express my gratitude to Mr. Phang Keat Keong and Mr. Lim Shiau Hong. Their advice and guidance are very valuables and useful to complete this project

I would also like to express my deepest appreciation to my family members for their continuous support and understanding. Last and not least, I would like to thanks my project members, Mr. Jimmy Tan Kai Hong, Mr. Tay Boon Pin, Mr. Sin Wai Kit, Mr. Wong Wing Hong, Mr. Yu Soon Lye, Mr. Phung Jacen and Ms. Wong Chee Sum for their support and cooperation.

TABLE OF CONTENT

ABSTRACT I

ACKNOWLEDGEMENT II

TABLE OF CONTENT III

LIST OF FIGURES VI

LIST OF TABLES VII

CHAPTER 1: Introduction 1

1.1 Introduction to Networking Technologies 1

1.1.1 Introduction of Ethernet 2

1.1.2 Introduction of ATM..... 2

1.1.3 Introduction of LANE 3

1.2 Introduction to Network Simulator 4

1.3 Project Objectives 4

1.4 Goals of Project..... 5

1.5 Project Schedule..... 5

1.6 Report Organization 6

CHAPTER 2: Literature Review..... 7

2.1 Review of Networking Technologies..... 7

2.1.1 An Introduction on ATM 7

2.1.1.1 ATM Basics..... 8

2.1.1.2 ATM Reference Model 8

2.1.1.3 ATM Logical Connections..... 10

2.1.1.4 ATM Cells..... 10

2.1.1.5 ATM Classes of Services 12

2.1.2 IP Over ATM 14

2.1.2.1 Classical IP over ATM..... 15

2.1.2.2 LANE 15

2.1.2.3 MPOA 21

2.2 Introduction to Computer Simulation 22

2.2.1 Network Simulation Approaches 23

2.2.2 Study of Various Existing Simulator 23

2.2.2.1 INSANE 24

2.2.2.2 NIST ATM/HFC 25

2.2.2.3 YATS 26

2.2.2.4 OMNeT++ 27

2.2.2.5 NetSim++ 27

2.2.2.6 Comparison 28

2.3 Programming Approaches..... 28

2.3.1 Procedural Approach..... 29

2.3.2 Structured Programming Approach..... 29

2.3.3 Object Oriented Approach..... 29

2.4 Programming Language 31

2.4.1 C++ 31

2.4.2 Java..... 32

2.4.3 Comparison of Java and C++ 34

2.5 Summary 34

CHAPTER 3: LANE	35
3.1 LANE Operation	35
3.1.1 Initialization and Configuration	36
3.1.2 Joining and Registering with the LES	37
3.1.3 Finding and Joining the BUS	37
3.1.4 Data Transfer	38
3.1.4.1 Resolution of IP Address to ATM Address	39
3.1.5 Managing ATM Broadcast	40
3.1.6 Distributed LAN Emulation	42
3.1.6.1 Single Server LANE Services Model	42
3.1.6.2 Distributed LAN Emulation Model	44
3.2.6.3 DLE ELAN	47
3.2 ELAN Access Control	49
3.3 Configuring ELAN	50
3.3.1 Configuring an LECS Configuration Database File	50
3.3.2 Defining an ELAN	51
3.3.3 Starting the LECs and Joining an ELAN	51
3.4 Summary	51
CHAPTER 4: Simulation Overview and Design	52
4.1 Overview of Simulation Model	52
4.1.1 Object Oriented Programming Approach	52
4.1.2 Multithreading	53
4.1.3 Portable	53
4.1.4 Programming Language Used	53
4.1.5 Programming Tools	54
4.2 JavaSim Network Simulator	55
4.2.1 Overview of JavaSim Network Simulator	55
4.2.2 JavaSim	56
4.2.3 SimClock	56
4.2.4 SimEvent	56
4.2.5 SimComponent	57
4.2.6 SimParameter	58
4.2.7 Object Serialization and Load/Save Function	59
4.3 Design of Components in the LANE	59
4.3.1 LECS	59
4.3.2 LES/BUS	61
4.3.3 LEC	63
4.4 Additional Parameter Design	64
4.4.1 SimParamLTable	64
4.4.2 SimParamLECache	64
4.4.3 SimParamMAC	65
4.4.4 SimParamIP	65
4.4.5 SimParamIntM	65
4.4.6 SimParamString	65
4.5 User Interface Design	65
4.6 Summary	66

CHAPTER 5: Simulator Implementation.....	67
5.1 Implementation.....	67
5.1.1 Control Frame class.....	67
5.1.2 LECS.....	68
5.1.2.1 LECS GUI Implementation.....	69
5.1.3 LES/BUS.....	69
5.1.3.1 LES/BUS GUI Implementation.....	70
5.1.4 LEC.....	70
5.1.4.1 GUI Implementation of LEC.....	73
5.1.5 SimParamLTable.....	73
5.1.6 SimParamLECache.....	74
5.2 Conclusion.....	75
CHAPTER 6: Testing.....	76
6.1 Component Testing.....	76
6.1.1 LANE Component Testing.....	76
6.1.1.2 Testing Result.....	76
6.1.2 SimParamLTable Component Testing.....	77
6.1.2.1 Testing Result.....	77
6.1.3 SimParamLECache Component Testing.....	78
6.1.3.1 Testing Result.....	79
6.2 Module Testing.....	80
6.2.1 Configuration Function Testing.....	80
6.2.2.1 Testing Result.....	81
6.2.2 Registration Function Testing.....	82
6.2.2.1 Testing Result.....	82
6.2.3 Data Transfer Function Testing.....	83
6.2.3.1 Testing Result.....	83
6.3 System Testing.....	84
6.3.1 Parameter Configuration of LANE Component.....	85
6.3.1.1 LE Configuration Server.....	85
6.3.1.2 LES.....	86
6.3.1.3 LECs.....	87
6.3.2 Simulation Testing.....	88
6.3.2.1 Configuration Phase.....	88
6.3.2.2 Join Phase.....	89
6.3.2.3 Data Transfer Phase.....	90
6.4 Summary.....	92
CHAPTER 7: Conclusion.....	93
7.1 System Strengths.....	93
7.2 System Limitation.....	94
7.3 Future Enhancement.....	94
REFERENCES.....	95
APPENDIX.....	97

LIST OF FIGURES

Figure 2.1:	The ATM reference model and OSI reference model	9
Figure 2.2:	ATM connection relationships	10
Figure 2.3:	ATM basic cell format, UNI cell and ATM NNI cell.....	11
Figure 2.4:	ATM bit rate services.....	13
Figure 2.5:	ATM network can emulate a physical LAN.....	16
Figure 2.6:	LANE protocol architecture.....	17
Figure 2.7:	An ELAN consists of clients, servers, and various intermediate nodes	18
Figure 2.8:	LANE data connections use s a series of VCCs to link LAN components	21
Figure 2.9:	LANE control connections link	21
Figure 2.10:	Three Sub-Fields of Computer Simulation.....	23
Figure 3.1:	Initial LEC configuration through the LECS.....	37
Figure 3.2:	Joining an emulated LAN	38
Figure 3.3:	Broadcast and multicast message managed by the BUS	41
Figure 3.4:	Single server LANE services model.....	42
Figure 3.5:	Broadcast IP-ARP request	43
Figure 3.6:	IP-ARP response handling	43
Figure 3.7:	Distributed LAN emulation model	44
Figure 3.8:	IP ARP broadcast from LEC 1 to LEC 9	45
Figure 3.9:	Re-distributing the broadcast across DLE peer servers.....	45
Figure 3.10:	LE-ARP response delivered and LEC 9 contacts LEC 1	46
Figure 3.11:	Registration on an ELAN with multiple servers	47
Figure 3.12:	ELAN with multiple server in operation	48
Figure 3.13:	Failure of one ELAN server and the recovery process.....	48
Figure 3.14:	ELAN re-established using the second server.....	49
Figure 4.1:	Hierarchy of Objects in the Simulator	55
Figure 6.1:	Simulation topology.....	85
Figure 6.2:	Simulation result of configuration phase	89
Figure 6.3:	LE Cache of comp LES.....	90

LIST OF TABLES

Table 1.1: Gantt chart for system schedule 5

Table 2.1: Service classification for ATM classes..... 13

Table 2.2: Comparison among various simulators..... 28

Table 6.1: LANE Component testing result..... 76

Table 6.2: SimParamLtable testing result 78

Table 6.3: SimParamLECache testing result..... 79

Table 6.4: LANE Database Configuration 80

Table 6.5: Configuration function testing result 81

Table 6.6: Configuration of LES/BUS 82

Table 6.7: Registration function testing result 83

Table 6.8: Data transfer function testing result..... 84

Table 6.9: Data Transfer of d1 90

The purpose of this course is to provide students with a comprehensive understanding of the principles and practices of the field. The course is designed to equip students with the necessary knowledge and skills to excel in their studies and future careers. The course is divided into several modules, each focusing on a specific aspect of the field. The first module, which is the focus of this chapter, introduces the students to the course and the program. The second module covers the fundamentals of the field, while the third module explores the advanced concepts and techniques. The fourth module focuses on the practical application of the knowledge and skills acquired during the course. The fifth module provides an overview of the current trends and developments in the field. The sixth module discusses the ethical and professional responsibilities of the students. The seventh module covers the career opportunities and the importance of continuous learning. The eighth module provides a summary of the course and the program. The ninth module discusses the importance of teamwork and communication skills. The tenth module provides a final assessment of the students' learning outcomes.

CHAPTER 1

INTRODUCTION

The purpose of this chapter is to provide students with a comprehensive understanding of the principles and practices of the field. The chapter is designed to equip students with the necessary knowledge and skills to excel in their studies and future careers. The chapter is divided into several sections, each focusing on a specific aspect of the field. The first section, which is the focus of this chapter, introduces the students to the course and the program. The second section covers the fundamentals of the field, while the third section explores the advanced concepts and techniques. The fourth section focuses on the practical application of the knowledge and skills acquired during the course. The fifth section provides an overview of the current trends and developments in the field. The sixth section discusses the ethical and professional responsibilities of the students. The seventh section covers the career opportunities and the importance of continuous learning. The eighth section provides a summary of the course and the program. The ninth section discusses the importance of teamwork and communication skills. The tenth section provides a final assessment of the students' learning outcomes.

The purpose of this chapter is to provide students with a comprehensive understanding of the principles and practices of the field. The chapter is designed to equip students with the necessary knowledge and skills to excel in their studies and future careers. The chapter is divided into several sections, each focusing on a specific aspect of the field. The first section, which is the focus of this chapter, introduces the students to the course and the program. The second section covers the fundamentals of the field, while the third section explores the advanced concepts and techniques. The fourth section focuses on the practical application of the knowledge and skills acquired during the course. The fifth section provides an overview of the current trends and developments in the field. The sixth section discusses the ethical and professional responsibilities of the students. The seventh section covers the career opportunities and the importance of continuous learning. The eighth section provides a summary of the course and the program. The ninth section discusses the importance of teamwork and communication skills. The tenth section provides a final assessment of the students' learning outcomes.

The purpose of this chapter is to provide students with a comprehensive understanding of the principles and practices of the field. The chapter is designed to equip students with the necessary knowledge and skills to excel in their studies and future careers. The chapter is divided into several sections, each focusing on a specific aspect of the field. The first section, which is the focus of this chapter, introduces the students to the course and the program. The second section covers the fundamentals of the field, while the third section explores the advanced concepts and techniques. The fourth section focuses on the practical application of the knowledge and skills acquired during the course. The fifth section provides an overview of the current trends and developments in the field. The sixth section discusses the ethical and professional responsibilities of the students. The seventh section covers the career opportunities and the importance of continuous learning. The eighth section provides a summary of the course and the program. The ninth section discusses the importance of teamwork and communication skills. The tenth section provides a final assessment of the students' learning outcomes.

CHAPTER 1: Introduction

1.1 Introduction to Networking Technologies

Changes in the structure of the telecommunications industry and market condition have brought new opportunities and challenges for network operator and public service providers to upgrade their existing network. LAN network such as Ethernet and Token Ring that have been primarily focused on providing basic connectivity services: connecting personal computer and terminals to mainframe and midrange systems that ran corporate applications, and providing workgroup connectivity at the departmental or division levels need to evolve in order to meet new multimedia communications challenges. Besides that, the current network need to support the increasing volume of data to be handled by the LANs due to the explosive growth in speed and computing power of personal computer.

Asynchronous Transfer Mode (ATM) inherent capabilities – such as gigabit-level speed, multi-services integration, virtual network and easy scalability make it an attractive alternative for network growth. IS professional are looking ways to upgrade their building, campus and departmental networks incrementally to ATM in order to control cost and migrate risk. They want to reap the benefits of high speed, high capacity ATM while preserving the best element in their existing network infrastructure. Their primary consideration is the interoperability of ATM technology with their installed base of Ethernet and Token Ring equipment, data networking protocols and legacy applications.

This is a valid concern, since ATM architecture is differs fundamentally from legacy LAN technologies. ATM is a connection-oriented technology. It uses abbreviated address, called a virtual channel identifier, to exchange data between two ATM stations over a virtual channel connection, VCC. Legacy LANs, on the other hand, employ connection less transmission technology based on global addressing. Most of today's data networking protocols have been designed to operate such connectionless networks. Thus, to use ATM for practical networking there must be someway of adapting existing network layer protocol, such as IP and IPX, to the connection –oriented paradigm of ATM.

Local Area Network Emulation (LANE) is the solution that provides a scalable migration path of legacy LAN to ATM. According to Version 1.0 of the ATM Forum LANE

specification, "The main objective of the LAN Emulation service is to enable existing applications to access an ATM network via protocol stacks like APPN, Net Bios, IPX, etc. as if they are running over traditional LANs." ANE works at the media access control (MAC) layer and enables legacy Ethernet, Token Ring, or FFDI traffic to run over ATM with no modifications to applications, network operating system, or desktop adapters. Legacy end stations can use LANE to connect to other legacy systems, as well as to ATM-attached servers, routers, hubs, and other networking devices.

1.1.1 Introduction of Ethernet

Within the IEEE 802 LAN standard committee, the 802.3 group has issued a set of standards with a common medium access control technique known as carrier-sense multiple access with collision detection (CSMA/CD). This set of standards grew out of the commercial product Ethernet, and the term Ethernet is still often used to refer all the specifications. Collectively, the Ethernet-like LANs are the dominant force in the LAN market.

The original base band version of CSMA/CD was developed by Xerox as part of the Ethernet LAN. With CSMA/CD, a station first senses the medium access and transmits only if the medium is idle. The station ceases transmission if it detects a collision. The advantage of CSMA/CD is its simplicity. It is easy to implement the logic required for this protocol. Furthermore, there is little to go wrong in the execution of the protocol.

Ethernet is connectionless transmission technology, which means no connection is setup between sender and receiver during the transmission of data.

1.1.2 Introduction of ATM

Asynchronous transfer mode (ATM) also known as cell-relay is a technology that has its history in the development of broadband ISDN in the 1970s and 1980s. ATM is perhaps the most important technical innovation to come out of the standardization work on broadband ISDN (B-ISDN). Cell-relay is similar in concept to frame relay. Both frame relay and cell relay take advantage of the reliability and fidelity of modern digital facilities to provide faster packet switching than X.25.

Cell relay is even more streamlined than frame relay in its functionality and can support data rates several orders of magnitude greater than frame relay. Both ITU-T and ATM forum has been developed the specifications for ATM.

Technically, ATM can be viewed as an evolution of packet switching. Like packet switching and frame relay, it allows multiple logical connections to be multiplexed over a single physical interface. ATM allows communication between devices that operate at different speeds and it is designed for high-performance multimedia networking.

The most basic service building block is the ATM virtual circuit, which is an end-to-end connection that has defined end points and routes but does not have bandwidth dedicated to it. Bandwidth is allocated on demand by the network as user has traffic to transmit. ATM defines various classes of service to meet a broad range of application needs.

1.1.3 Introduction of LANE

ATM Forum has created a specification for the coexistence of legacy LANs and ATM LANs, known as ATM LAN emulation. The objective of LAN Emulation is to enable existing shared media LAN nodes to interoperate across an ATM network and to interoperate across an ATM network with devices that connect directly to ATM switches.

ATM LAN emulation defines the way which end systems on two separate LANs of the same type (same MAC layer) can exchange MAC frames across the ATM network. Besides that, it defines the way in which an end system on a LAN can interoperate with an end system emulating the same LAN type and attached directly to an ATM switch.

ATM LAN emulation requires two types of components: clients and servers. Clients operate on behalf of devices that are attached to legacy LANs and that use of MAC address. A client is responsible of adding its MAC entities into the overall configuration and for dealing with the tasks associated with translating between MAC address and ATM address. Servers are responsible for integrating MAC entities into the overall configuration and for managing all the associated tasks, such as finding addresses and emulating broadcasting. Each emulated LANs consists of one or more client and a single LAN emulation service.

1.2 Introduction to Network Simulator

With the rapid development of high-speed network, network simulator has become a valuable tool to study and investigate the protocol and design issues regarding the performance of the network. It allow user to make correct decision on designing a network without the need to invest into the technology.

A network simulator can be used as a tool for ATM network planning or as a tool for ATM protocol performance analysis. It is useful for modeling network behavior under different setting and conditions for the various network components. Users are able to analyze and predict the performance of the network design based on the generated result of network simulator. Besides that, researchers and network planners are able to analyze networks without the expense of building a real network with the use of a simulator. Huge saving can be made both in terms of investment and the cost in terms of unnecessary restructuring for experimentation.

1.3 Project Objectives

The primary objective of this project is to study and understand the operation of LAN Emulation in ATM. This involves researching work done on LAN emulation and ATM as well as legacy LAN such as Ethernet. It reveals how ATM is configured to simulate a legacy LAN and co-exist with it. Besides that, a study on the simulation of LAN emulation is performed. This is needed to provide a basic idea in building a new LAN emulation simulator.

The second objective is the development of a new LAN emulation simulator based on several important features that differentiate it from current existing simulator. These features include platform independent, object-oriented programming approach that make use of expendability and reusability, multithreaded operation in which the objects run in parallel. Besides that, this simulator can be used as a training kit for user in designing and testing LANE before implementing the real network.

1.4 Goals of Project

The ultimate goal of this project is to create the components needed by the ATM LAN emulation simulator to properly simulate the operation in LANE. The components developed are able to integrate into a complete ATM network simulator.

The ATM LAN emulation simulator should have the following capabilities:

- Allow the initialization and configuration of a new LEC
- Allow the creation of virtual LANs in the emulated LAN
- Extend the broadcast capability of legacy LAN functioning in LAN emulation
- Allow data transfer to operate in emulate LAN

1.5 Project Schedule

In the purpose of achieving the objectives of this system, a milestone of the whole system is drawn. The milestone will arrange the time for each stage of the system development and leads to preparing a guideline in developing the system. Below is the milestone for the system in details:

Table 1.1: Gantt chart for system schedule

Task	Jul 00	Aug 00	Sep 00	Oct 00	Nov 00	Dec 00	Jan 01
Literature Review							
System Analysis							
System Design							
System Coding/Implementation							
System Testing/Evaluation							
Documentation							

1.6 Report Organization

The subsequent chapters of this report is organized in the following manner:

Chapter 1 – Introduction: This chapter provides an introduction to the current networking technologies associates with this project, project objectives, goals of projects and report organization.

Chapter 2 – Literature Review: This chapter covers the literature review and research work done during the project. The chapter mainly covers 3 sections. The first section is a review of networking technologies that is ATM and LAN emulation. The second section covers the evaluation of current existing network simulator and the last section reviews the programming approaches that can be used to develop the simulator.

Chapter 3 – LANE: This chapter explains in details the configuration and operation in LANE.

Chapter 4 – Simulation Overview and Design: This chapter gives an overview of the analysis and requirement to developed the network simulator. Besides that, it covers the design aspects of the LAN emulation simulator. This includes the design of object classes for component in LANE such as LEC, LECS, BUS and LES. Besides that, it details their tasks and uses, as well as interaction between object classes.

Chapter 5 – Simulator Implementation: This chapter covers the implementation details for the ATM LANE Network Simulator components. It explains the major attributes and methods of each class.

Chapter 6 – Testing: This chapter covers the component testing, modules testing and system testing to ensure the simulator runs correctly.

Chapter 7 – Conclusion: This chapter concludes the work on the development of ATM LANE simulator component. It summarizes the findings of this project and the outcome of the development process.

CHAPTER 2: Literature Review

This chapter will provide the background and context for the research project. It will also provide a summary of the current state of knowledge on the topic and identify the research gaps that the project aims to address.

CHAPTER 2

LITERATURE REVIEW

The literature review is a critical component of any research project. It provides a comprehensive overview of the current state of knowledge on a particular topic, identifies research gaps, and informs the development of the research question and objectives. The literature review is a critical component of any research project. It provides a comprehensive overview of the current state of knowledge on a particular topic, identifies research gaps, and informs the development of the research question and objectives.

2.1 Introduction to the Literature Review

The literature review is a critical component of any research project. It provides a comprehensive overview of the current state of knowledge on a particular topic, identifies research gaps, and informs the development of the research question and objectives.

2.1.1 The Importance of the Literature Review

The literature review is a critical component of any research project. It provides a comprehensive overview of the current state of knowledge on a particular topic, identifies research gaps, and informs the development of the research question and objectives.

The literature review is a critical component of any research project. It provides a comprehensive overview of the current state of knowledge on a particular topic, identifies research gaps, and informs the development of the research question and objectives.

CHAPTER 2: Literature Review

This chapter details the literature review work done throughout the duration of the project. This chapter is broken into four main sections. This chapter has covered the research background of this project and provides relevant knowledge to this project.

The first section of this project covered the review of networking technologies that will be simulated. The result of the review technologies has provided the characteristic to be implemented to the network simulator.

The second chapter covered the computer simulation and analyzes the network simulation available in the market. The analysis of current available network simulator reveals their strength and weakness. Thus, The strength of the simulator can be incorporate into the network simulator.

The third sections discuss the various programming approach that can be used to develop the network simulator. The following section covers the programming languages that can be used to develop the network simulator. The comparison of each programming languages is done in order to choose the appropriate programming languages that meets the requirement to develop the network simulator.

2.1 Review of Networking Technologies

The network technologies of ATM and LAN emulation features allow them to be useful means for data communications. This section provide a details review on these technologies.

2.1.1 An Introduction on ATM

Asynchronous Transfer Mode (ATM) is an International Telecommunication Union-Telecommunication Standardization Sector (ITU-T) standard for cell relay wherein information for multiple service types, such as voice, video, or data is conveyed in small, fixed size cells [1]. ATM networks are connection oriented.

ATM is based on the efforts of the ITU-T Broadband Intergraded Services Digital Network (BISDN) standard. The ATM Forum extended the ITU-T's vision on of ATM for use over public and private networks.

2.1.1.1 ATM Basics

ATM is a cell-switching and multiplexing technology that combines the benefits of circuit switching (guaranteed capacity and constant transmission delay) with those of packet switching (flexibility and efficiency for intermittent traffic). It provides scalable bandwidth from a few megabits per second (Mbps) to many gigabits per second (Gbps). Because of its asynchronous nature, time slots are available on demand with information identifying the source of the transmission contained in the header of each ATM cell.

ATM transfers information in fixed-size units called cells. Each cell consists of 53 octets, or bytes. The first 5 bytes contain cell-header information, and the remaining 48 contain the “payload” (user information). Small fixed-length cells are well suited to transferring voice and video traffic because such traffic is intolerant of delays that result from having to wait for a large data packet to download, among other things.

ATM networks are fundamentally connection oriented, which means that a virtual channel (VC) must be set up across the ATM network prior to any data transfer. When an ATM device wants to establish a connection with another ATM device, it sends a signaling-request packet to its directly connected ATM switch. This request contains the ATM address of the desired ATM endpoint, as well as any QoS parameters required for the connection.

2.1.1.2 ATM Reference Model

ATM is a streamlined protocol with minimum error and flow control capabilities, this reduces the overhead of processing ATM cells and reduces the number of overhead bits required with each cells, thus simplifies the processing required at each ATM node, again supporting the use of ATM at high data rates [2].

The standards issued for ATM by ITU-T are based on the protocol architecture shown in Figure 2.1, which illustrates the basic architecture for an interface between user and network. The physical layer involves the specification of a transmission medium and a signal-encoding scheme. It controls transmission and receipt of bits on the physical medium. It keeps track of ATM cell boundaries and package cells into appropriate type of frame for the physical medium being used. The ATM physical layer is divided into two parts, which are physical medium sublayer and transmission convergence sublayer.

Two layer of the protocol architecture relate to ATM functions. There is an ATM layer common to all services that provide packet transfer capabilities, and an ATM adaptation layer (AAL) that is service dependent. The ATM layer defines the transmission of data in fixed-size cells and the use of logical connections. The use of ATM creates the need for an adaptation layer to support information transfer protocols not based on ATM. The AAL maps higher-layer information into ATM cells to be transported over an ATM network, and then collects information from ATM cells for delivery to higher layer. Several ATM adaptations layers are currently specified.

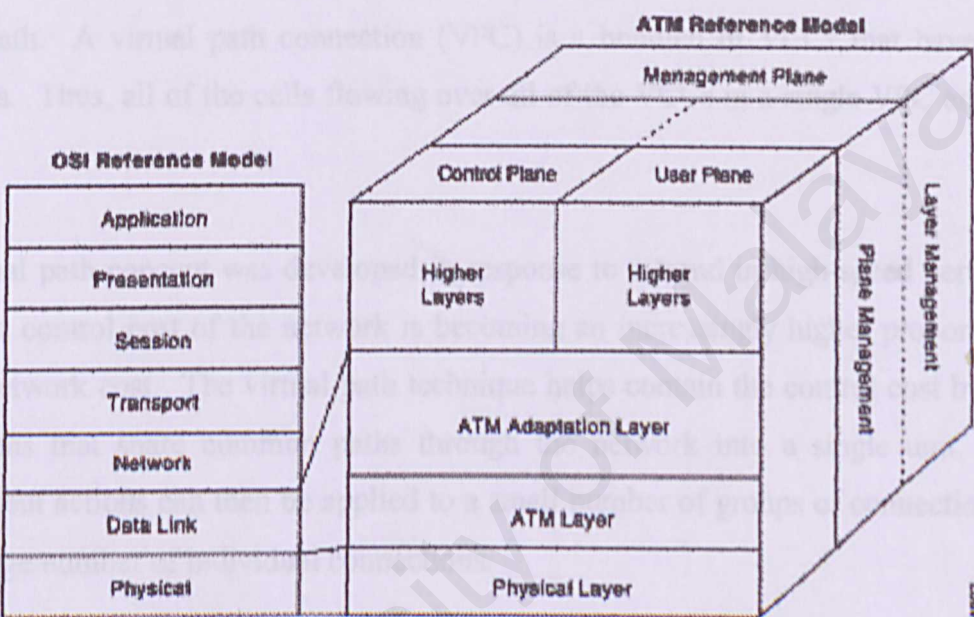


Figure 2.1: The ATM reference model and OSI reference model

The protocol reference model makes reference to three separate planes:

- User Plane – provides for user information transfer, along with associated controls (e.g., flow control, error control)
- Control Plane – performs call control and connection control functions
- Management Plane – includes plane management, which performs management functions related to a system as a whole and provides coordination between all the planes, and layer management that perform management functions relating to resources and parameters residing in its protocol entities.

2.1.1.3 ATM Logical Connections

Logical connections in ATM are referred to as virtual channel connections (VCC). A VCC is analogous to virtual circuit in X.25 or data link connection in frame relay; it is the basic unit in an ATM network. A VCC is set up between two end users through the network and a variable-rate, full duplex flow of fixed-size cells is exchanged over the connection. VCCs are used for user-network exchange (control signaling) and network-network exchange (network management and routing).

For ATM, a second sublayer of processing has been introduced that deals with the concept of virtual path. A virtual path connection (VPC) is a bundled of VCCs that have the same endpoints. Thus, all of the cells flowing over all of the VCCs in a single VPC are switched together.

The virtual path concept was developed in response to a trend in high-speed networking in which the control cost of the network is becoming an increasingly higher proportion of the overall network cost. The virtual path technique helps contain the control cost by grouping connections that share common paths through the network into a single unit. Network management actions can then be applied to a small number of groups of connections instead of to a large number of individual connections.

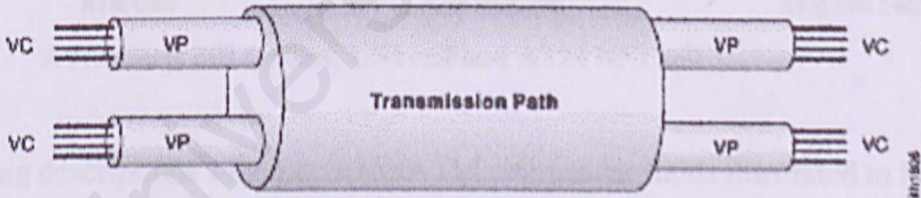


Figure 2.2: ATM connection relationships

2.1.1.4 ATM Cells

The asynchronous transfer mode makes use of fixed-size cells, which consists of a 5-octet header and a 48-octet information field. There are several advantages to the use of small, fixed-size cells. First, their use may reduce queuing delay for a high-priority cell, as it waits less if it arrives slightly behind a lower-priority cell that has gained access to resource. Secondly, it appears that fixed-size cells can be switched more efficiently, which is important

for the very high data rates of ATM. With fixed-size cells, it is easier to implement the switching mechanism in hardware.

Figure 2.3 illustrate the basic ATM cell format, the ATM UNI cell-header format, and the ATM NNI cell-header format. Unlike the UNI, the NNI header does not include the Generic Flow Control (GFC) field. Additionally, the NNI header has a Virtual Path Identifier (VPI) field that occupies the first 12 bits, allowing for larger trunks between public ATM switches.

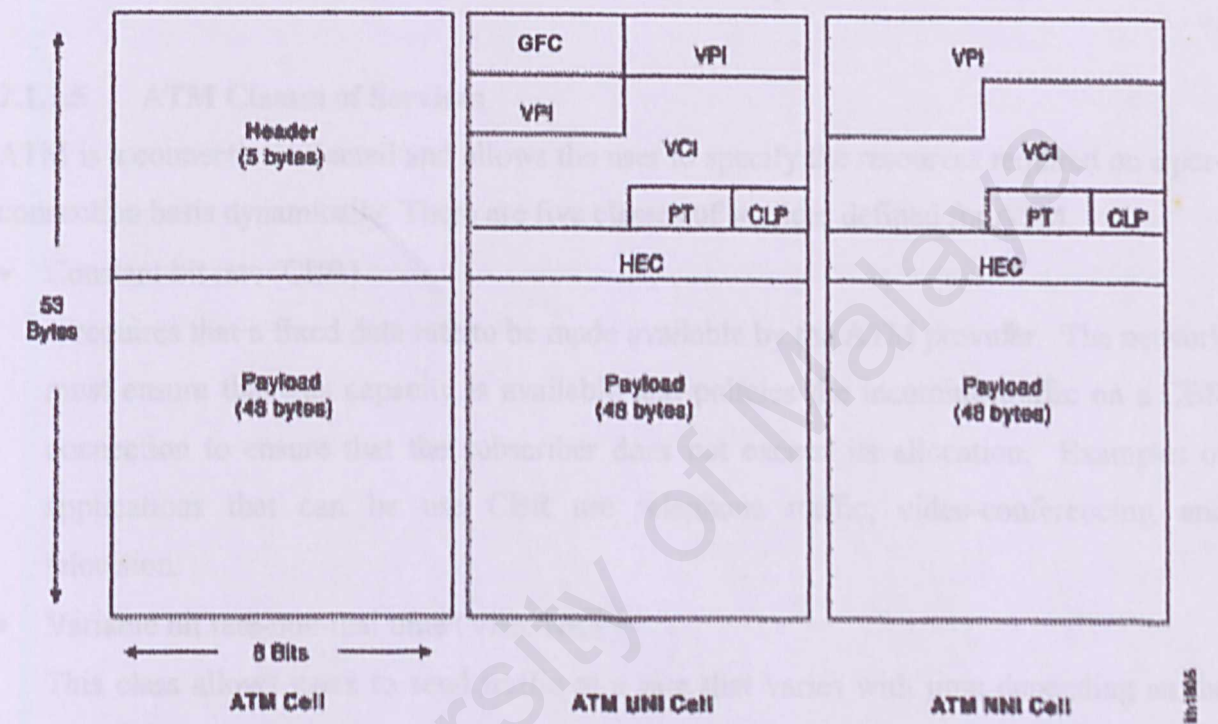


Figure 2.3: ATM basic cell format, UNI cell and ATM NNI cell

The following descriptions summarize the ATM cell-header fields illustrated in Figure 2.3:

- Generic Flow Control (GFC) - Provides local functions, such as identifying multiple stations that share a single ATM interface. This field is typically not used and is set to its default value.
- Virtual Path Identifier (VPI) - In conjunction with the VCI, identifies the next destination of a cell as it passes through a series of ATM switches on the way to its destination.
- Virtual Channel Identifier (VCI) - In conjunction with the VPI, identifies the next destination of a cell as it passes through a series of ATM switches on the way to its destination.

- **Payload Type (PT)** - Indicates in the first bit whether the cell contains user data or control data. If the cell contains user data, the second bit indicates congestion, and the third bit indicates whether the cell is the last in a series of cells that represent a single AAL5 frame.
- **Congestion Loss Priority (CLP)** - Indicates whether the cell should be discarded if it encounters extreme congestion as it moves through the network. If the CLP bit equals 1, the cell should be discarded in preference to cells with the CLP bit equal to zero.
- **Header Error Control (HEC)** - Calculates checksum only on the header itself.

2.1.1.5 ATM Classes of Services

ATM is a connection oriented and allows the user to specify the resources required on a per-connection basis dynamically. There are five classes of services defined for ATM.

- **Constant bit rate (CBR)**
It requires that a fixed data rate to be made available by the ATM provider. The network must ensure that this capacity is available and policies the incoming traffic on a CBR connection to ensure that the subscriber does not exceed its allocation. Examples of applications that can be use CBR are telephone traffic, video-conferencing, and television.
- **Variable bit rate-non-real time (VBR-NRT)**
This class allows users to send traffic at a rate that varies with time depending on the availability of user information. Statistical multiplexing is provided to make optimum use of network resources.
- **Variable bit rate-real time (VBR-RT)**
This class is similar to VBR-NRT but is designed for applications that are sensitive to cell-delay variation. Examples for real-time VBR are voice with speech activity detection (SAD) and interactive compressed video.
- **Available bit rate (ABR)**
This class of services provides rate-based flow control and is aimed at data traffic such as file transfer and e-mail. Although the standard does not required the cell transfer delay and cell-loss ratio to be guaranteed or minimized, it is desirable for switches to minimize delay and loss as much as possible. Depending upon the state of congestion in the network, the source is required to control its rate. The users are allowed to declare a minimum cell rate, which is guaranteed to the connection by the network.

- Unspecified bit rate (UBR)
This class provides best effort service. No amount of capacity is guaranteed and cells may be discarded.

Table 2.1 shows the service classifications of ATM classes. CBR provides a service similar to a leased line of a dial-up switched circuit. A constant bit rate is maintained. VBR is useful for applications that run at an uneven rate but require a given capacity on demand. ABR and UBR support LAN internetworking and other types of data traffic. Typically, data traffic is much burstier than voice or video traffic. A constant or nearly constant delivery rate is not required. For the user, the concern is one of throughput, while for the network the concern is that bursts of traffic from many users at the same time could overwhelm switches, causing cells to be dropped. ABR is intended for applications in which delay is a concern, such as on-line sessions between a user and a server. UBR is directed at delay-tolerant applications, such as file transfer and electronic mail.

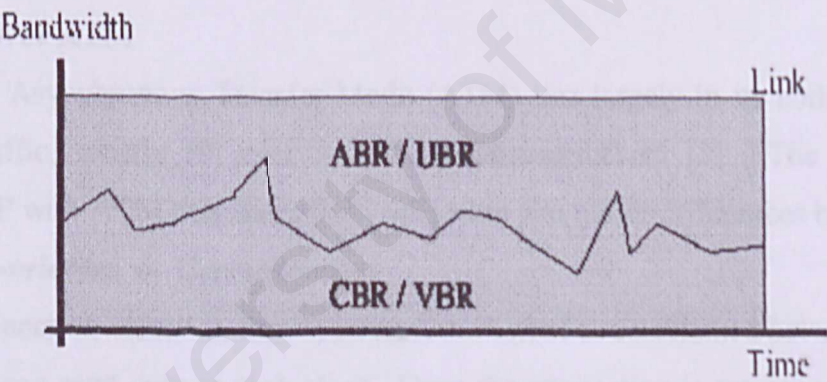


Figure 2.4: ATM bit rate services

Table 2.1: Service classification for ATM classes

ATM services	CBR	VBR-RT	VBR-NRT	ABR & UBR
Timing relation between source and destination	Required		Not required	
Bit rate	Constant	Variable		
Connection mode	Connection oriented			Connectionless

There are QoS parameters correspond to the network performance at the ATM layer that defines the ATM service categories. Those negotiated parameters are as follows:

- Cell lost ratio (CLR) – CLR is the percentage of cells not delivered at their destination because they were lost in the network due to congestion and buffer overflow.
- Cell transfer delay (CTD) – the delay experienced by a cell between network entry and exit points is called CTD. It includes propagation delays, queuing delays at various intermediate switches, and service times at queuing points.
- Cell delay variation (CDV) – CDV is a measure of the variance of the cell transfer delay. High variation implies larger buffering for delay-sensitive traffic such as voice and video.
- Peak cell rate (PCR) – The maximum cell rate at which the user will transmit. PCR is the inverse of minimum cell inter-arrival time.
- Sustained cell rate (SCR) – this is the average rate, as measured over a long interval, in the order of the connection lifetime.
- Burst tolerance (BT) – this parameter determines the maximum burst that can be sent at the peak rate. This is the bucket-size parameter for the enforcement algorithm that is used to control the traffic entering the network.

2.1.2 IP Over ATM

The success of Asynchronous Transfer Mode (ATM) lies largely in its ability to transport legacy data traffic, mostly IP, over its network infrastructure [3]. The complexity of interoperating IP with ATM originates from following two major differences between them.

- *Connection-oriented vs. Connectionless*

ATM is connection-oriented, that is, a connection needs to be established between two parties before they can send data to each other. Once the connection is set up, all data between them is sent along the connection path. On the contrary, IP is connectionless so that no connection is needed and each IP packet is forwarded by routers independently on a hop-by-hop basis.

- *QoS-aware vs. Best Effort*

Quality of Service is an important concept in ATM networks. It includes the parameters like the bandwidth and delay requirements of a connection. Such requirements are included in the signaling messages used to establish a connection. Current IP (IPv4) has no such concept and each packet is forwarded on a best effort basis by the routers.

In order to run IP on top of ATM networks, it needs to relate ATM protocol layers to TCP/IP layers. Two models, one called peer model and other overlay model are proposed.

Peer model considers the ATM layer a peer networking layer as IP and propose the use of the same addressing scheme as IP for ATM-attached end system. ATM signaling request contain IP address and the intermediate switches will route the request using existing routing protocols. This scheme was rejected because although it simplifies the addressing scheme for end system, it complicates the design of ATM switches by requiring them to have all the functions of an IP router. Moreover, if the ATM network can support other networking layer protocols like IPX or AppleTalk, the switch has to understand all their routing protocols.

The overlay model, which is finally adopted, views ATM as a data link layer protocol on top of which IP runs. In overlay model ATM networks have its addressing scheme and routing protocols. The ATM address space is not logically coupled with the IP addressing space and there will be no arithmetic mapping between the two. Each end system will typically has an ATM address and an unrelated IP address as well. Since there is no nature mapping between the two addresses, the only way to figure out one is through some addressing resolution protocol.

2.1.2.1 Classical IP over ATM

With overlay model, there are essentially two ways to run IP over ATM. One treats ATM as a LAN and partition an ATM network into several logical subnets consisting of end systems with the same IP prefix. This is known as Classical IP over ATM [4]. In Classical IP over ATM, end system in the same logical subnet communicate with each other through end-to-end ATM connections, and like in LAN, ARP servers are used in logical subnets to resolve the IP addresses into ATM addresses. However, traffic between two end systems in different subnets has to go through a router even though they are attached to the same ATM network. This is not desirable since routers introduce high latency and become the bandwidth bottleneck. Next Hop Resolution protocol (NHRP) [5] steps in to solve this problem. Working in an ATM network partitioned into logical subnets, it allows an end system in one subnet to resolve the ATM address (from the IP address) of an end system in another logical subnets and establish an end-to-end ATM connection, called a short cut, between them.

2.1.2.2 LANE

LANE is a standard defined by the ATM Forum [6] that gives to stations attached via ATM the same capabilities they normally obtain from legacy LANs, such as Ethernet and Token

Ring. As the name suggests, the function of the LANE protocol is to emulate a LAN on top of an ATM network. Specifically, the LANE protocol defines mechanisms for emulating either an IEEE 802.3 Ethernet or an 802.5 Token Ring LAN. The current LANE protocol does not define a separate encapsulation for FDDI. (FDDI packets must be mapped into either Ethernet or Token Ring emulated LANs [ELANs] by using existing translational bridging techniques. Fast Ethernet (100BaseT) and IEEE 802.12 (100VG-AnyLAN) both can be mapped unchanged because they use the same packet formats. Figure 2.5 compares a physical LAN and an ELAN.

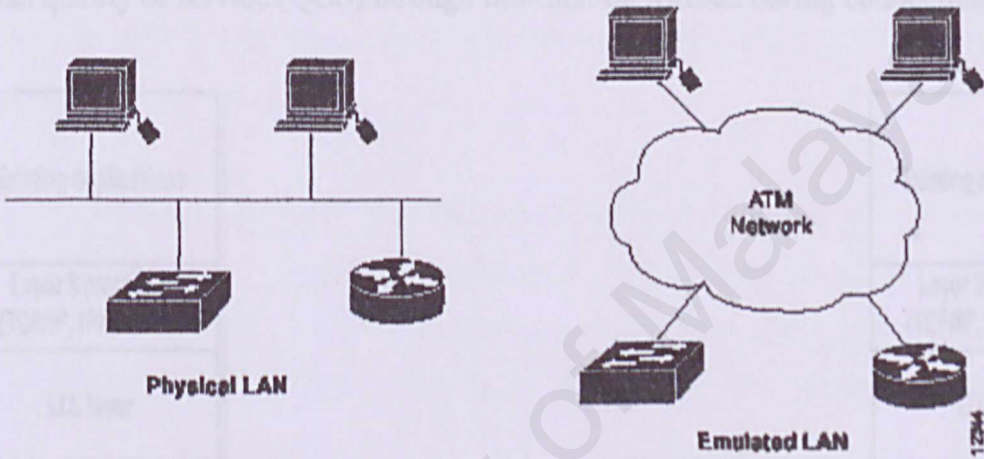


Figure 2.5: ATM network can emulate a physical LAN

The LANE protocol defines a service interface for higher-layer (that is, network layer) protocols that is identical to that of existing LANs. Data sent across the ATM network is encapsulated in the appropriate LAN MAC packet format. Simply put, the LANE protocols make an ATM network look and behave like an Ethernet or Token Ring LAN—albeit one operating much faster than an actual Ethernet or Token Ring LAN network.

It is important to note that LANE does not attempt to emulate the actual MAC protocol of the specific LAN concerned (that is, CSMA/CD for Ethernet or token passing for IEEE 802.5). LANE requires no modifications to higher-layer protocols to enable their operation over an ATM network. Because the LANE service presents the same service interface of existing MAC protocols to network-layer drivers (such as an NDIS- or ODI-like driver interface), no changes are required in those drivers.

2.1.2.2.1 LANE Protocol Architecture

LANE provides a translation layer between the higher-level connectionless protocols and the lower-level connection-oriented ATM protocols, as shown in Figure 2.6. Consider the protocol layer differences between the ATM host at the figure's far left and the Ethernet or Token Ring host at the far right. The ATM layer manages the header for the 53-byte ATM cell. It accepts the cell payload from a higher layer, appends the header, and passes the resultant fixed-length cell to the physical layer below. Conversely, it receives cells from the physical layer, strips off the header, and passes the remaining 48 bytes to the higher layer protocols. The ATM layer is unaware of the types of traffic it carries, although it can distinguish quality of service (QOS) through information learned during connection setup.

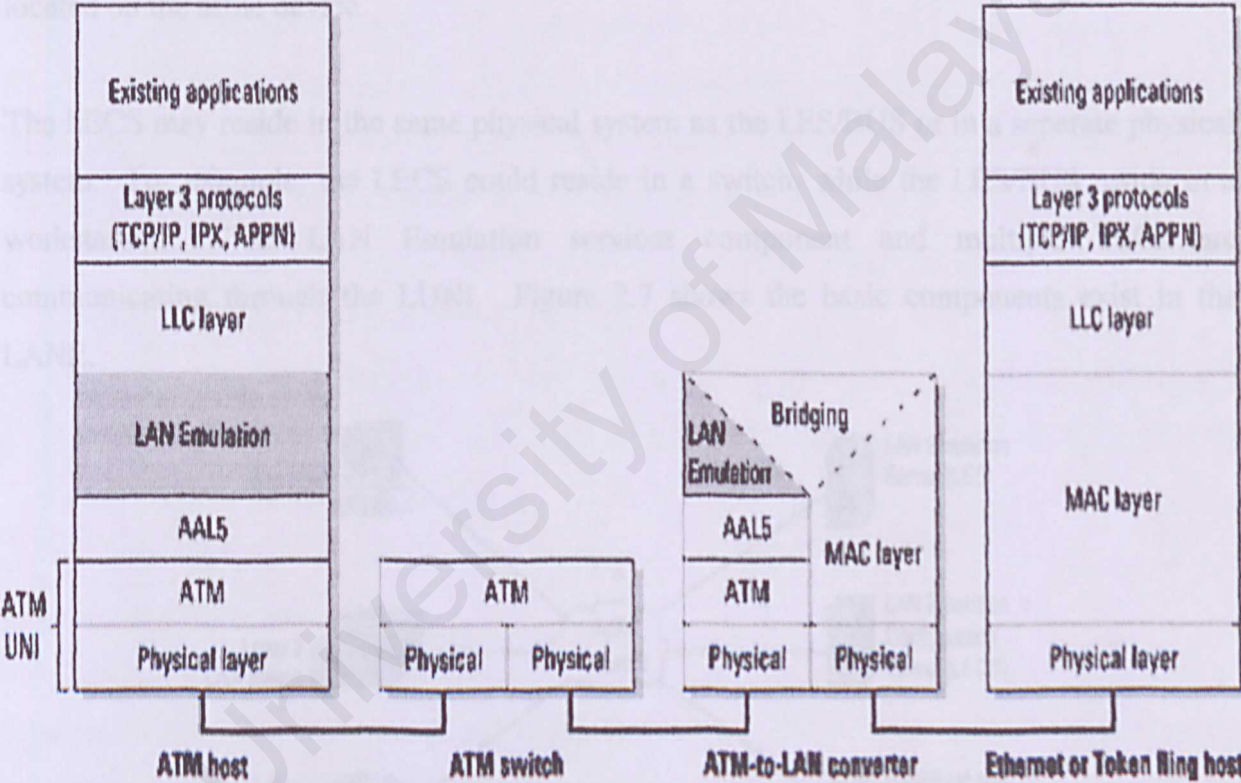


Figure 2.6: LANE protocol architecture

The ATM adaptation layer (AAL) sits above the ATM layer. The AAL formats data into the 48-byte ATM cell payload, a process known as segmentation. Once the ATM cells reach their destination, they are reconstructed into higher-level data and transmitted to the respective local devices, a process called reassembly. Because ATM can carry multiple traffic types, several adaptation protocols, each operating simultaneously, can exist at the adaptation layer. AAL Type 5 is used for LAN Emulation.

LANE sits above AAL5 in the protocol hierarchy. It masks the connection setup and handshaking functions required by the ATM network from the higher protocol layers and thus are completely independent of upper-layer protocols, services, and applications. Conversely, it maps the MAC address-based data networking protocols into ATM virtual connections so that the higher-layer protocols think they are operating on a connectionless LAN.

2.1.2.2.2 LANE Components

The components of an ELAN include LECs, and LAN Emulation services consisting of a LECS, a LES and a BUS. Although the ATM Forum specification allows the LES and BUS to be located on different devices, more intelligent traffic handling is possible when they are located on the same device.

The LECS may reside in the same physical system as the LES/BUS or in a separate physical system. For example, the LECS could reside in a switch, while the LES/BUS reside in a workstation. The LAN Emulation services component and multiple LECs are communicating through the LUNI. Figure 2.7 shows the basic components exist in the LANE.

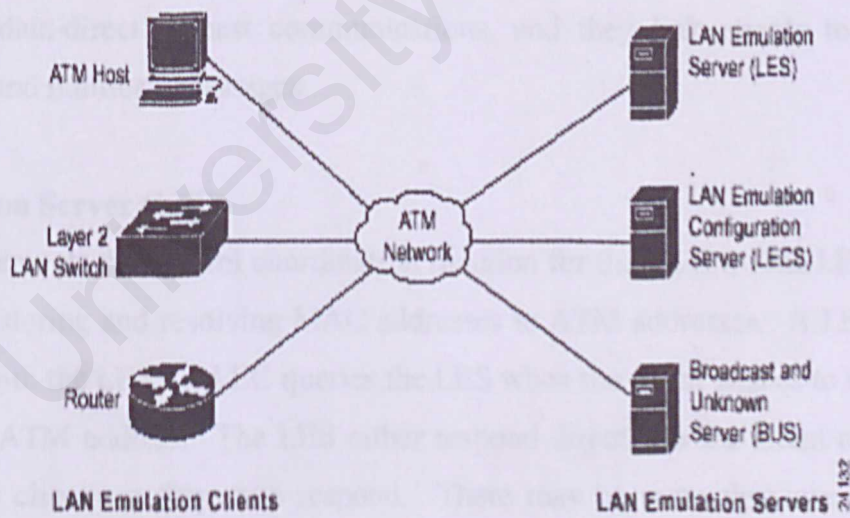


Figure 2.7: An ELAN consists of clients, servers, and various intermediate nodes

LAN Emulation Client (LEC)

The LEC is the component in an end system that performs data forwarding, address resolution, and other control functions when communicating with other components within

the ELAN. It provides a MAC level emulated Ethernet or Token Ring interface and appears to higher level software as though a physical interface is present. Each LEC must register with both the LES and BUS associated with the ELAN it wishes to join before it may participate in the ELAN. To participate in multiple ELANs, an end system must have multiple LECs. Currently, up to 16 LECs are supported on adapter cards running Solaris or Windows/NT software, and up to 4 LECs on adapter cards running Windows 95.

LAN Emulation Configuration Server

The LECS is responsible for dynamically assigning different LECs to different emulated LANs. It provides the clients with the address of the most appropriate LES and maintains a database of the resultant associations. It can assign a LEC to an emulated LAN based on either physical location, as specified by the LEC's ATM address, or by logical association. A single LECS can manage the configuration information for a very large ATM network, since its responsibilities are limited to initial configuration. LECs communicate with the LAN Emulation service functions through two different types of VCCs:

- Control connections carry administrative messages, such as requests for initial configuration and for addresses of other LECs.
- Data connections handle all other communications. In particular, they link clients to each other for data-direct unicast communications, and they link clients to the BUS for broadcast and multicast messages.

LAN Emulation Server (LES)

The LES implements the control coordination function for the ELAN. The LES provides the service of registering and resolving MAC addresses to ATM addresses. A LEC registers its own address with the LES. A LEC queries the LES when the client wishes to resolve a MAC address to an ATM address. The LES either respond directly to the client or forwards the query to other clients so they map respond. There may be more than one instance of an active LES per ELAN.

Broadcast and Unknown Server (BUS)

Unlike traditional shared-media LAN architectures such as Ethernet and Token Ring, ATM is connection based. Therefore, it has no built in mechanism for handling connectionless traffic such as broadcast, multicast and unknown unicast. In an ELAN, the BUS is responsible for

servicing these traffic types by accepting broadcast, multicast, and unknown unicast packets from the LECs via dedicated point-to-point connections, and forwarding the packets to all of the members of the ELAN using a single point-to-multipoint connection. Unknown unicast packets are packets that the sending station broadcast because it does not yet know the ATM address for the packet's destination MAC address. There may be more than one instance of an active BUS per ELAN.

2.1.2.2.3 LAN Emulation Connection Types

The LANE entities communicate with each other by using a series of ATM VCCs. LECs maintain separate connections for data transmission and control traffic. The LANE data connections are data-direct VCC, multicast send VCC, and multicast forward VCC.

Data-direct VCC is a bidirectional point-to-point VCC set up between two LECs that want to exchange data. Two LECs typically use the same data-direct VCC to carry all packets between them rather than opening a new VCC for each MAC address pair. This technique conserves connection resources and connection setup latency.

Multicast send VCC is a bidirectional point-to-point VCC set up by the LEC to the BUS. Multicast forward VCC is a unidirectional VCC set up to the LEC from the BUS. It typically is a point-to-multipoint connection, with each LEC as a leaf. Figure 2.8 shows the LANE data connections.

Control connections include configuration-direct VCC, control-direct VCC, and control-distribute VCC. Configuration-direct VCC is a bi-directional point-to-point VCC set up by the LEC to the LECS. Control-direct VCC is a bi-directional VCC set up by the LEC to the LES. Control-distribute VCC is a unidirectional VCC set up from the LES back to the LEC (this is typically a point-to-multipoint connection). Figure 2.9 illustrates LANE control connections.

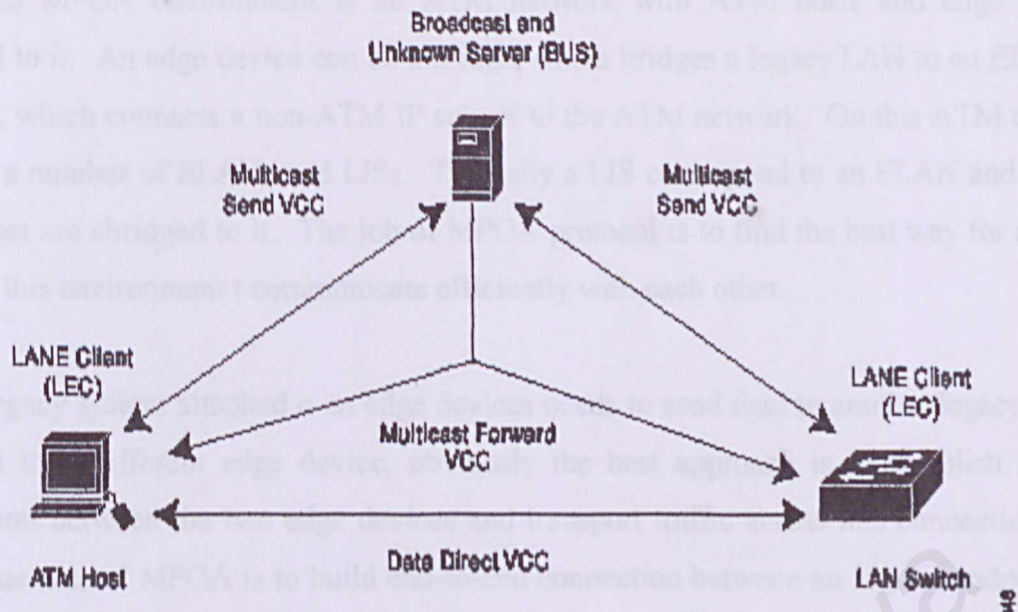


Figure 2.8: LANE data connections use s a series of VCCs to link LAN components

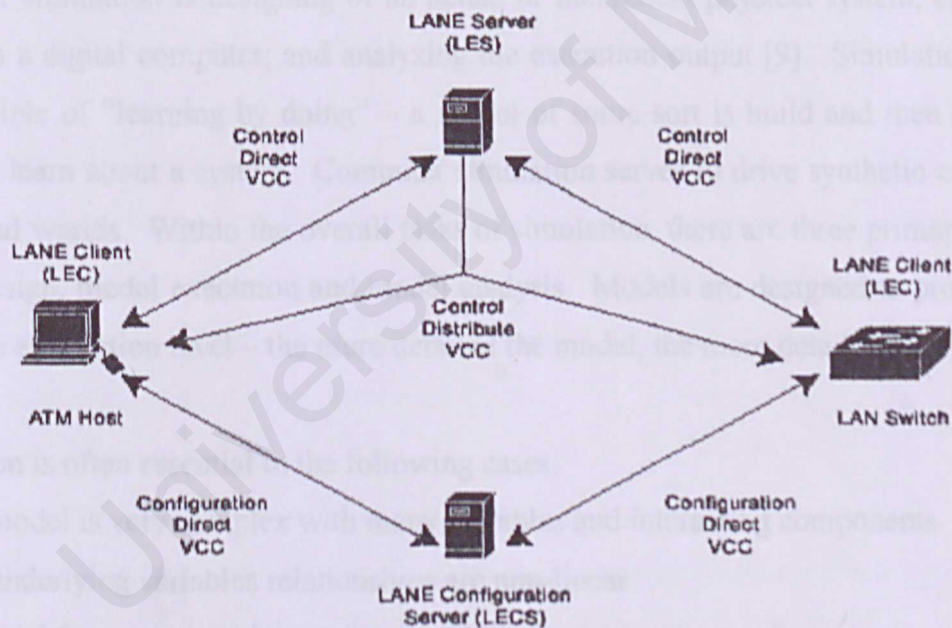


Figure 2.9: LANE control connections link

2.1.2.3 MPOA

Multiprotocol [7] over ATM is basically a combination of LANE and NHRP. MPOA improves LANE by allowing inter-ELAN traffic go through shortcuts connections rather than routers. The build such a shortcut, NHRP is used to resolve destination IP address into ATM address. In this sense, MPOA is a combination of Layer 3 routing and Layer 2 bridging [8].

A typical MPOA environment is an ATM network with ATM hosts and edge devices attached to it. An edge device can be a bridge, which bridges a legacy LAN to an ELAN, or a router, which connects a non-ATM IP subnet to the ATM network. On this ATM network defined a number of ELANs and LISs. Typically a LIS correspond to an ELAN and Legacy LANs that are abridged to it. The job of MPOA protocol is to find the best way for any two hosts in this environment to communicate efficiently with each other.

If one legacy system attached to an edge device needs to send data to another legacy system attached to a different edge device, obviously the best approach is to establish a direct connection between the two edge devices and transport traffic across this connection. The major business of MPOA is to build end-to-end connection between an ingress endpoint and egress endpoint for efficient communication.

2.2 Introduction to Computer Simulation

Computer simulation is designing of an actual or theoretical physical system, executing the model on a digital computer, and analyzing the execution output [9]. Simulation embodies the principle of “learning by doing” – a model of some sort is build and then operates the model to learn about a system. Computer simulation serves to drive synthetic environments and virtual worlds. Within the overall tasks of simulation, there are three primary sub-fields: model design, model execution and model analysis. Models are designed to provide answer at a given abstraction level – the more detailed the model, the more detail the output.

Simulation is often essential in the following cases:

- The model is very complex with many variables and interacting components
- The underlying variables relationships are non-linear
- The model contains random variates
- The model output is to be visual as in a 3D computer animation

The power of simulation is that – even for easily solvable linear systems – a uniform model execution technique can be used to solve a large variety of systems. Another important aspects of the simulation technique are to builds a simulation model to replicate the actual system.

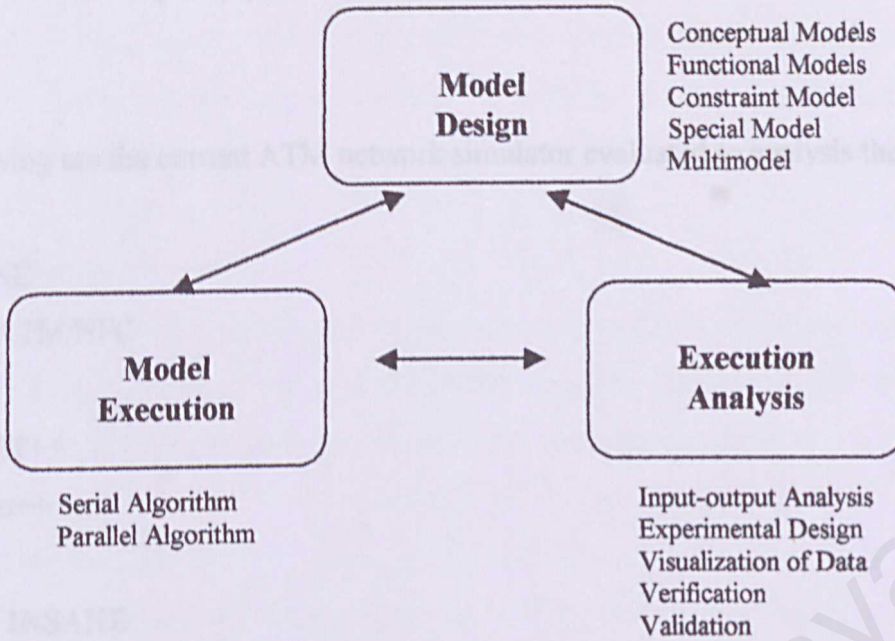


Figure 2.10: Three Sub-Fields of Computer Simulation

2.2.1 Network Simulation Approaches

There are two approaches to modeling a network simulator. These two approaches are as below:

- Analytical modeling**
 Analytical modeling is a closed form approach of network modeling method which the network is model mathematically in the form of equation. The main disadvantage of analytical models is over simplistic view of the network and their inability to simulate the dynamic nature of a computer network.
- Discrete event modeling**
 The computer replicates the real world objects, which means the objects play certain roles and changes its state at a discrete point during simulation. This approach is more accurate but it requires more modeling time in developing the system. Besides that, it need more time in processing the real world objects.

2.2.2 Study of Various Existing Simulator

A network simulator is used to perform experiments on network without the expanses of building a real network. It help user to perform analysis on the network and obtain accurate information in order to plan and design the network more efficiently. Generally, ATM network simulators are able to support network performance analysis in varying traffic types

and loads, network capacity planning, traffic aggregation studies and ATM network protocol research.

The following are the current ATM network simulator evaluated to analysis their strength and weakness:

- INSANE
- NIST ATM/HFC
- YATS
- OMNET++
- NetSim++

2.2.2.1 INSANE

INSANE (Internet Simulated ATM Networking Environment) is designed to test various IP-over-ATM algorithms with realistic traffic loads derived from empirical traffic measurements [10]. INSANE's ATM protocol stack provides real-time guarantees to ATM virtual circuits by using Rate Controlled Static Priority (RCSP) queuing. ATM signaling is performed using a protocol similar to the Real-Time Channel Administration Protocol (RCAP). Internet protocols supported include large subsets of IP, TCP, and UDP. In particular, the simulated TCP implementation performs connection management, slow start, flow and congestion control, retransmission, and fast retransmits. Various application simulators mimic the behavior of standard Internet applications to provide a realistic workload, including: telnet, ftp, WWW, real-time audio, and real-time video.

INSANE is designed to run large simulations whose results are processed off-line. It works quite well on distributed computing clusters (although simulations are all sequential processes, a large number of them can easily be run in parallel).

Although there is no graphical user interface, a (optional) Tk-based graphical simulation monitor provides an easy way to check the progress of multiple running simulation processes. The bulk of INSANE is written in C++. Customization and simulation configuration is performed with Tcl scripts

Advantages

The Tk-based graphical simulation monitor enable user to check the progress of multiple running simulation process. Besides that, it is able to support the simulation on a large network, which the result is processed off-line.

Disadvantages

The simulator can only works on a few hardware and platforms only and this restricted the portability of the simulator. Furthermore, there are a few software requirements to run the simulator and this will be troublesome for the user to use the software.

2.2.2.2 NIST ATM/HFC

This simulator was developed at the National Institute of Standards and Technology (NIST) and it is a tool to analyze the behavior of ATM and HFC networks without the expense of building a real network [11]. Therefore, this simulator can conceivably be used to plan be used to plan ATM networks as well as analyze ATM and HFC protocols. It allows the user to interactively model the environment with a graphical user interface.

By using the NIST ATM/HFC simulator, the user can create different network topologies; adjust the parameters of each component's operation, measure network activity, save/load different simulation configuration and log data simulation execution.

Advantages

The user can create different topologies and able to adjust the parameters during the simulation of the network. The user can save and load various simulation configuration. The simulator provides a graphical user interface and enable user to drag and drop the entities in the network.

Disadvantages

Users of the simulator might face problems setting up the network topology because they need to input a large number of parameters. The customization of the simulator's component requires user or programmers to have strong foundation in C. Besides that, it is using procedural approach whereby the components have overlapped functions between the

components. The simulator only can run on limited platform that is UNIX and LINUX platform.

2.2.2.3 YATS

YATS (Yet Another Tiny Simulator) is a small cell-level simulation tool for ATM. Its kernel comprises the event scheduler, a symbol manager and a scanner/parser front end [12]. An input file describes the - arbitrary - model network configuration, the simulation actions and the way to analyze the results. The input language is a simple script language, which allows for a flexible problem description (loops, macros and basic mathematical capabilities are provided). The discrete-time event scheduler applies a static calendar queue and unusual event memory management, which results in good simulation speed. The system is written in C++. All network nodes are objects that communicate over standardized messages. Graphical object classes are able to display the time dependent behavior of variables and distributions inside of other model objects (without adding complexity to these network objects).

Advantages

The simulation of the network has reasonable speed and simple models virtually can run in real time. The simulator has high flexibility of integrated model description, simulation control, and result analysis. The whole simulation experiment can be instrumented via environment variables that in turn allows - together with a shell script - to easily perform complete experiment series over night. Although it is very simple, the online displays are useful to understand what's happening in the model network. This especially holds for ABR, TCP and all this protocol stuff.

Disadvantages

The pure slotted operation causes some restriction when simulating different line speeds in the same model. It's only possible to choose speeds for which the cell transfer time is an integer multiple of a basic time used for the whole model. Lower line speeds are emulated by the multiplexer objects classes MuxAF/MuxDF, the ABR multiplexer does not yet support lower speeds. The discrete-time nature excludes some useful source models like the Poisson types. While the language based model description yields a high flexibility, the input may become a bit irritating in case of larger networks.

2.2.2.4 OMNeT++

OMNeT++ (Objective Modular Network Testbed in C++) is a discrete event simulation tool [13]. It is primarily designed to simulate computer networks, multi-processors and other distributed systems, but it may be useful for modeling other systems tool. OMNeT++ has been developed on Linux, but it works just as well on most Unix systems and on Windows platforms (NT recommended). It provides a simulation library with statistical classes and an environment that supports interactive simulation including the visualization of collected data. The gnu plot-based GUI tool is used for analyzing and plotting simulation results.

Advantages

OMNeT++ has a solid and flexible simulation kernel and it provide powerful GUI environment for simulation execution. Users can build hierarchical and reusable models easily. The interface is human readable and its source code is provided.

Disadvantages

User must use command line to simulate the network and posses knowledge in C or C++ programming languages to use OMNet++.

2.2.2.5 NetSim++

In a nutshell, NetSim++ is a software package designed to provide a comprehensive work environment for the network modeler. It can be used in areas of communications networks such as performance measurement for existing or future networks under a wide range of conditions [14]. Besides that, it can perform analysis and simulation of queuing systems.

NetSim++ is designed specifically for the development and analysis of communications networks. Models can be hierarchically structured, allowing their re-use in different simulations. Specifications are entered graphically with specialized editors. The editors provide an efficient medium for design capture via a consistent set of modern user-interface elements. NetSim++ follows for the hierarchy and communication model a subset of SDL-92 semantics. As with SDL, the active parts are processes; a hybrid approach is used to embed C++ language code with a graphically specified Extended Finite State Machine (EFSM).

Advantages

NetSim++ provides an efficient event-driven Simulation Kernel, a Simulation API and a Base Models Library of components. It takes the design specification and automatically generates an executable simulation. A set of analysis tools is provided to interpret and visualize a large volume of simulation results.

Disadvantages

The current implementation of NetSim++ is available only for UNIX/X Window System platforms.

2.2.2.6 Comparison

Based on the evaluation of the network simulator, the Table 2.2 below compare the network simulator on a few feature such as discrete-event simulator, object-oriented, GUI, multithreaded, web enabled, platform independent.

Table 2.2: Comparison among various simulators

Simulator	Discrete event simulation	Object-oriented	GUI	Multithread	Web-enable	Platform independent
INSANE	√	√	√	√	X	X
NIST ATM/HFC	√	X	√	X	X	X
YATS	√	√	X	√	X	X
OMNet++	√	√	√	X	X	X
NetSIM++	√	X	√	√	X	X

2.3 Programming Approaches

There are several programming approaches to develop an ATM network simulator. These include procedural approach, structured approach and object-oriented approach that are widely used in developing a network simulator. This section discusses these three approaches.

2.3.1 Procedural Approach

In procedural approach, the program codes are placed into blocks that are referred as procedures or functions. A function or procedure is a relatively simple program that is called by other programs and returns a value to the program that called it. With the use of procedural approach, the task were broken down into separate blocks, in which separate blocks would perform separate tasks. Computer languages like Pascal, C and FORTRAN are procedural programming languages.

In a procedural-based programming language, a programmer writes out instructions that are followed by a computer from start to finish.

2.3.2 Structured Programming Approach

The principle idea behind structured programming is as simple as the idea of divide and conquers. A computer program can be thought of as consisting a set of tasks. Any task that is too complex to be described simply would be broken down into a set of smaller component tasks, until the tasks were sufficiently small and self-contained enough that they were easily understood.

As an example, computing the average salary of every employee of a company is a rather complex task. However, it can be break down into these subtasks:

- *Find out what each person earns.*
- *Count how many people in the company.*
- *Total all the salaries.*
- *Divide the total by the number of people.*

Structured programming remains an enormously successful approach for dealing with complex problems. By the late 1980s, however, some of the deficiencies of structured programming had become all too clear.

2.3.3 Object Oriented Approach

A revolutionary concept that changed the rules in computer program development, object-oriented programming (OOP) is organized around object rather than actions, data rather than logic. Historically, a program has been viewed as a logical procedure that takes input data,

processes it, and produces output data. The programming challenge was seen as how to write the logic, not how to define the data. Objects are essentially reusable software components that model items in the real world. Software developers are discovering that using a modular, object-oriented design and implementation approach can make software developments group much more productive than is possible with previous popular programming techniques such as structured programming. Object oriented programs are easier to understand.

1.4.1 Programming Language

Object-oriented programming takes the view that what we really care about are the objects we want to manipulate rather than the logic required to manipulate them. Examples of objects range from human beings (described by name, address, and so forth) to buildings and floors (whose properties can be described and managed) down to the little widgets on your computer desktop (such as buttons and scroll bars).

1.4.2 Advantages of the network simulator

The first step in OOP is to identify all the objects to manipulate and how they relate to each other, an exercise often known as data modeling. Once an object is identified, it is generalized as a class of objects and define the kind of data it contains and any logic sequences that can manipulate it. The logic sequences are known as methods. A real instance of a class is called an "object" or, in some environments, an "instance of a class." Its methods provide computer instructions and the class object characteristics provide relevant data. Objects can communicate with each other - with well-defined interfaces called *messages*.

1.4.3 Advantages of OOP

The concepts and rules used in object-oriented programming provide these important benefits:

- The concept of a data class makes it possible to define subclasses of data objects that share some or all of the main class characteristics. Called inheritance, this property of OOP forces a more thorough data analysis, reduces development time, and ensures more accurate coding.
- Since a class defines only the data it needs to be concerned with, when an instance of that class (an object) is run, the code will not be able to accidentally access other program data. This characteristic of data hiding provides greater system security and avoids unintended data corruption.

- The definition of a class is re-useable not only by the program for which it is initially created but also by other object-oriented programs (and, for this reason, can be more easily distributed for use in networks).
- The concept of data classes allows a programmer to create new data types that are not defined in the language itself.

2.4 Programming Language

There are a few of programming languages that can be used in the development of the network simulator. In addition, the features of the programming language must be able to meet the requirements of the system to be developed. Since most of the simulator is built in Object Oriented Programming approach, the programming language must support the OOP approach. The choices of programming language should be able to support other functionality of the network simulator.

In this section, it reviews two main OOP programming languages, which are C++ and Java. These two programming language has become the most popular language used to develop the network simulator.

2.4.1 C++

C++ evolved from C, which evolved from two precious programming languages, BCPL and B. C++ was developed by Bjarne Stroustrup as an extension of C in early 1980s at Bell Laboratories. Because C++ retains C as a subset, it gains many of the attractive features of the C language, such as efficiency, closeness to the machine, and a variety of built-in types. A number of new features were added to C++ to make the language even more robust and more importantly, it provides capabilities for object-oriented programming.

Below are the features of C++:

- C++ is object-oriented – it allow user to develop software using object-oriented approach. User can create their objects that model the real world item and thus increase the reusability of the software.
- C++ supports inheritance and polymorphism - Inheritance is a form of software reusability in which new classes are created from existing classes by absorbing their attributes and behaviors and overriding or embellishing these with capabilities the new

classes require. Polymorphism enables users to write programs in general fashion to handle a wide variety of existing and yet-to-be-specified related classes. Inheritance and polymorphism are effective techniques for dealing with software complexity.

- C++ supports operator overloading – C++ enables user to overload most operators to be sensitive to the context in which they are used. The compiler generates the appropriate code based on the manner in which the operator is used.
- C++ is robust – it enable user to write clearer, more robust, more fault tolerant programs. User can write methods to deal with the unusual occurrences or exceptions.

2.4.2 Java

Java was developed at Sun Microsystems. Work on Java originally began with the goal of creating a platform-independent language and operating system for consumer electronics. The original intent was to use C++, but as work progressed in this direction, the Java developers realized that they would be better creating their own language rather than extending C++.

Java is both a programming language and an environment for executing programs written in the Java language. Unlike traditional compilers, which convert source code into machine-level instructions, the Java compiler translates Java source code into instructions that are interpreted by the runtime Java Virtual Machine. So, unlike languages like C and C++, on which Java is based, Java is an interpreted language.

Java is best described as a small, simple, safe, object-oriented, interpreted or dynamically optimized, byte-coded, architecture-neutral, garbage-collected, multithreaded programming language with a strongly typed exception-handling mechanism for writing distributed, dynamically extensible programs.

Java has several of important features that make it an attractive programming language as below:

- Java is simple – Java started out as C++ but has had certain features removed, it is certainly a simpler language than C++. Java has simplified C++ programming by both adding features beyond those found in C++ and by removing some of the features that make C++ a complicated and difficult language to master. Java is simple because it

consists of only three primitive data types-numbers, Boolean types, and arrays. Everything else in Java is a class.

- Java is object-oriented – The design of Java is completely object-oriented. Java provides all the luxuries of object-oriented programming: class hierarchy, inheritance, encapsulation, and polymorphism-in a context that is truly useful and efficient. Java's object-oriented structure enables user to develop more useful, more tailor able, and much simpler software the first time around.
- Java supports the Internet – Java can be used to build small application modules or applet for use as part of a Web page. Applets make it possible for a Web page user to interact with the page.
- Java is general purpose - Java can be used to create complete applications that may run on a single computer or be distributed among servers and clients in a network.
- Java is robust – The Java objects can contain no references to data external to themselves or other known objects. This ensures that an instruction cannot contain the address of data storage in another application or in the operating system itself, either of which would cause the program and perhaps the operating system itself to terminate or "crash". The Java virtual machine makes a number of checks on each object to ensure integrity.
- Java is secure - Closely related to Java's robustness is its focus on security. Because Java does not use pointers to directly reference memory locations, as is prevalent in C and C++, Java has a great deal of control over the code that exists within the Java environment.
- Java is platform-independent - The programs created are portability in a network. The program is compiled into Java byte code that can be run anywhere in a network on a server or client that has a Java. The Java virtual machine interprets the byte code into code that runs on the real computer hardware. This means that individual computer platform differences such as instruction lengths can be recognized and accommodated locally just as the program is being executed. Platform-specific versions of the program are no longer needed.
- Java supports multithreaded –Java support for multiple, synchronized threads that are built directly into the Java language and runtime environment. Synchronized threads are extremely useful in creating distributed, network-aware applications. Such an application may be communicating with a remote server in one thread while interacting with a user in a different thread.

2.4.3 Comparison of Java and C++

Java bears a strong relationship to the widely used C++ programming language. There are, however, a number of significant differences, the most significant of which include:

- Java is much more simple than C++ to develop an application. Although C++ is a powerful and very expressive language, it is difficult and programmer-unfriendly.
- Java is more portable compare to C++. Application developed using Java is portable and can run on any hardware or platform. C++ application needs to be customized in order to run on certain platforms.
- Java can be used to develop web-based application but C++ take longer time in developing a web-based application.
- Java is more robust than C++. Java objects can contain no references to data external or other known objects. This ensures that an instruction cannot contain the address of data storage in another application or in the operating system itself, either of which would cause the program and the operating system itself to terminate or "crash". As in C++, pointer arithmetic, casting and explicit memory management using "new" and "delete" are the major causes of bugs in C++ programs.

2.5 Summary

This chapter has covered the primary research background of this project and relevant knowledge needed to develop the network simulator. A more detailed explanation of LANE operation will be covered in the following chapter.

CHAPTER 3 LAN

The chapter highlights the main types of LANs and the factors that influence their performance. It also discusses the various LAN topologies and the factors that influence their performance. The chapter concludes with a discussion of the various LAN standards and the factors that influence their performance.

CHAPTER 3

LAN EMULATION

LAN and LAN emulation

LAN emulation is a technology that allows a network to emulate a LAN. It is used to connect different types of networks, such as token ring and Ethernet, and to provide a common interface for all the networks. LAN emulation is used in a variety of applications, including network management, network security, and network performance monitoring.

LAN emulation is a technology that allows a network to emulate a LAN. It is used to connect different types of networks, such as token ring and Ethernet, and to provide a common interface for all the networks. LAN emulation is used in a variety of applications, including network management, network security, and network performance monitoring.

LAN emulation is a technology that allows a network to emulate a LAN. It is used to connect different types of networks, such as token ring and Ethernet, and to provide a common interface for all the networks. LAN emulation is used in a variety of applications, including network management, network security, and network performance monitoring.

LAN emulation is a technology that allows a network to emulate a LAN. It is used to connect different types of networks, such as token ring and Ethernet, and to provide a common interface for all the networks. LAN emulation is used in a variety of applications, including network management, network security, and network performance monitoring.

CHAPTER 3: LANE

This chapter highlights the configuration and operation in LANE. In the first section, it presents a detailed description of LANE operation, which consists of four phases. The next section described the steps in configuring the LECS, LES/BUS and LEC.

3.1 LANE Operation

The operation of LANE system and component include 4 stages of LEC operations: initialization and configuration; joining and registering with the LES; finding and joining the BUS; and data transfer.

In the initialization and configuration phase, a LEC contacts the LECS to know each ELAN it belongs to and the address of LES and BUS to contact to join that ELAN. There are three ways for a LEC to access LECS. The first is to configure the ATM address of the LECS into LEC. The second is to have a fixed VPI/VCI that directs to the LECS from every system. The last is to get it through ILMI.

During joining and registration phase, the LEC knows the ATM address of the LES of the ELAN to join; it sends a registration message to the LES, which includes the MAC address and ATM address of the LEC. Upon receiving this message, the LES will record the information in its address resolution table and create a few connections to the LEC for transfer of data and control information.

LEC will use the ATM address of BUS obtained in the registration phase to establish a connection to BUS for the transfer of multicast data.

Data transfer phase is the major business of LANE. When a LEC wants to send some data to another LEC, it will query LES for the ATM address of that LEC. Using that address, it establishes a direct data connection with that LEC. Afterwards Ethernet frame between them are transported on this connection in AAL 5 frames with LLC encapsulation. If a LEC needs to broadcast a packet, it simply sends the packet to the BUS, which will forward every ELAN member a copy of the packet.

The section below will provide a more details explanation of the four major LANE operations.

3.1.1 Initialization and Configuration

When a LEC first powers up, it must obtain configuration information from the LECS in order to join an emulated LAN. The LANE specification offers several options for locating the LECS:

- The LEC can use a "well-known address," conceptually an address like 1-800-FIND-LECS. Theoretically, this method provides an elegant, easy-to-implement solution for configuring multiple LECs. However, the "well-known address" capability and the address format were defined in anticipation of future ATM signaling specifications, and therefore this is not an interoperable approach today; this issue will be addressed in a future version of the LANE specification.
- The LEC can send Interim Local Management Interface (ILMI) messages to its ATM switch. This alternative requires the network manager to configure the address of the LECS in each ATM switch in the network. When the LEC powers up, it would send ILMI messages across the UNI requesting a LECS address; the attached switch would then respond. Currently this is the most convenient method for discovering the LECS, since it requires the configuration of only ATM switches, not the hosts, bridges, routers, and other LEC devices on the network.
- The LEC could use a predefined VCC. This alternative requires a pre-configured VCC to the LECS from every ATM interface on the network—every port with a UNI. This approach requires a large number of VCCs, the majority of which will be idle most of the time.
- Finally, the LECS can be bypassed completely by configuring the ATM address of a LES in the LEC.

When the LECS is found, the LEC sets up a configuration-direct VCC to the LECS and sends a `LE_CONFIGURE_REQUEST`. If a matching entry is found, the LECS returns a `LE_CONFIGURE_RESPONSE` to the LEC with the configuration information it requires to connect to its target ELAN, including the following: ATM address of the LES, type of LAN being emulated, maximum packet size on the ELAN, and ELAN name (a text string for display purposes).

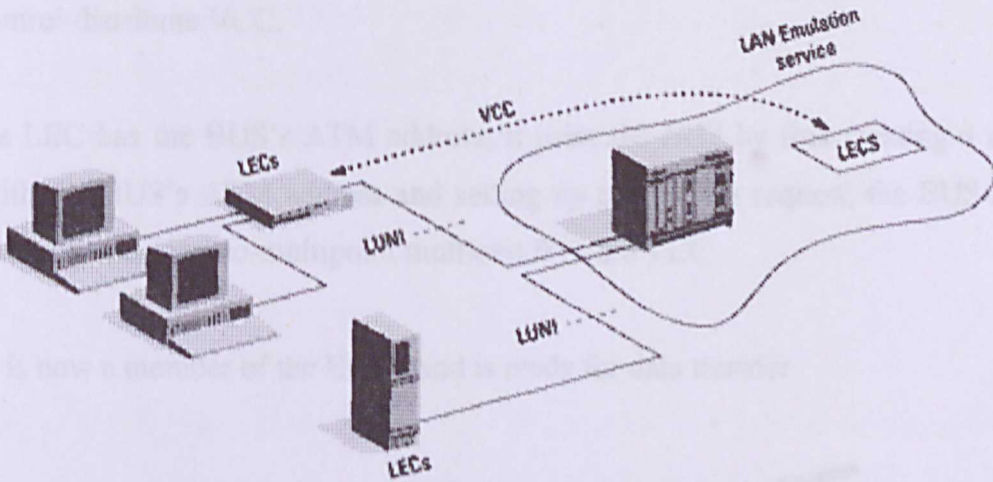


Figure 3.1: Initial LEC configuration through the LECS

3.1.2 Joining and Registering with the LES

When an LEC joins the LES and registers its own ATM and MAC addresses, it does so by following three steps:

- After the LEC obtains the LES address, the LEC optionally clears the connection to the LECS, sets up the control-direct VCC to the LES, and sends an LE_JOIN_REQUEST on that VCC. This allows the LEC to register its own MAC and ATM addresses with the LES and (optionally) any other MAC addresses for which it is proxying. This information is maintained so that no two LECs will register the same MAC or ATM address.
- After receipt of the LE_JOIN_REQUEST, the LES checks with the LECS via its open connection, verifies the request, and confirms the client's membership.
- Upon successful verification, the LES adds the LEC as a leaf of its point-to-multipoint control-distribute VCC and issues the LEC a successful LE_JOIN_RESPONSE that contains a unique LAN Emulation Client ID (LECID). The LECID is used by the LEC to filter its own broadcasts from the BUS.

8.1.3 Finding and Joining the BUS

After the LEC has successfully joined the LECS, its first task is to find the BUS/s ATM address to join the broadcast group and become a member of the emulated LAN. First, the LEC creates an LE_ARP_REQUEST packet with the MAC address 0xFFFFFFFF. Then the LEC sends this special LE_ARP packet on the control-direct VCC to the LES. The LES

recognizes that the LEC is looking for the BUS and responds with the BUS's ATM address on the control-distribute VCC.

When the LEC has the BUS's ATM address, it joins the BUS by first creating a signaling packet with the BUS's ATM address and setting up a signaling request, the BUS adds the LEC as a leaf on its point-to-multipoint multicast forward VCC.

The LEC is now a member of the ELAN and is ready for data transfer.

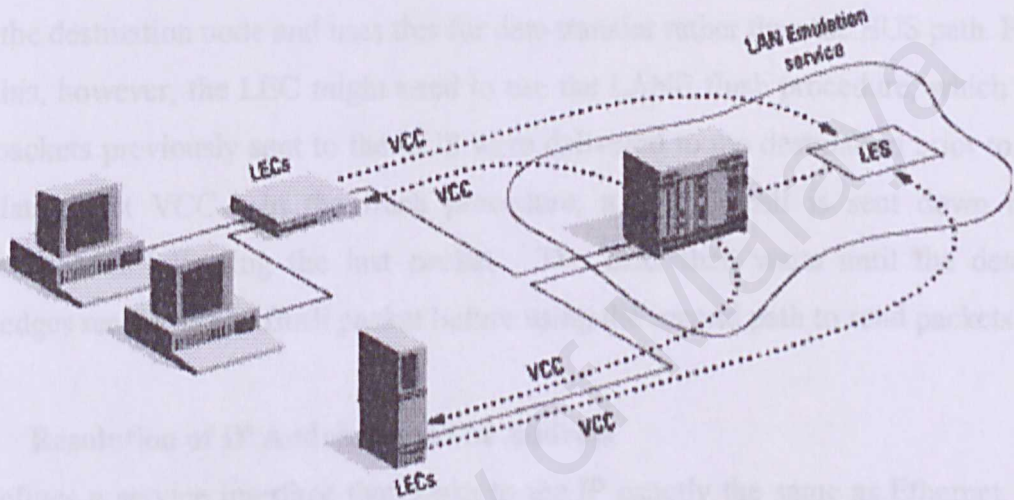


Figure 3.2: Joining an emulated LAN

3.1.4 Data Transfer

The final state, data transfer, involves resolving the ATM address of the destination LEC and actual data transfer, which might include the flush procedure.

When a LEC has a data packet to send to an unknown-destination MAC address, it must discover the ATM address of the destination LEC through which the particular address can be reached. To accomplish this, the LEC first sends the data frame to the BUS (via the multicast send VCC) for distribution to all LECs on the ELAN via the multicast forward VCC. This is done because resolving the ATM address might take some time, and many network protocols are intolerant of delays.

The LEC then sends a *LAN Emulation Address Resolution Protocol Request* (LE ARP Request) control frame to the LES via a control-direct VCC.

If the LES knows the answer, it responds with the ATM address of the LEC that owns the MAC address in question. If the LES does not know the answer, it floods the LE_ARP_REQUEST to some or all LECs (under rules that parallel the BUS's flooding of the actual data frame, but over control-direct and control-distribute VCCs instead of the multicast send or multicast forward VCCs used by the BUS). If bridge/switching devices with LEC software participating in the ELAN exist, they translate and forward the ARP on their LAN interfaces.

In the case of actual data transfer, if an LE_ARP is received, the LEC sets up a data-direct VCC to the destination node and uses this for data transfer rather than the BUS path. Before it can do this, however, the LEC might need to use the LANE flush procedure, which ensures that all packets previously sent to the BUS were delivered to the destination prior to the use of the data-direct VCC. In the flush procedure, a control cell is sent down the first transmission path following the last packet. The LEC then waits until the destination acknowledges receipt of the flush packet before using the second path to send packets.

3.1.4.1 Resolution of IP Address to ATM Address

LANE defines a service interface that looks to the IP exactly the same as Ethernet. In this way the IP software that is running previously on Ethernet can be ported onto ATM network without any modification. In this case, the mapping of IP address to ATM address will need 2 tables:

- **IP to MAC Table**

The IP address is mapped to MAC address as it was done in legacy LAN. LANE is not even aware that such a thing is done.

- **MAC to ATM Table**

The MAC address is mapped to ATM address as it was done in LANE in which the LEC will send an LE_ARP_REQUEST to the LES to resolve the ATM address of the destination LEC.

When a host sending a packet to a destination host, it leads to sending an IP packet. Sender host has an IP to MAC (ARP) table in which it does not find a MAC address for destination host IP. So it sends an ARP request with broadcast MAC address as the destination address. LEC sees that and sends it through the Multicast send VCC to the BUS.

BUS will receive the ARP request from the LEC and sends it to all the LECs in the ELAN. All the clients pick it up but ARP server of one client finds that someone is asking for his MAC address, so it coins ARP response and sends it down. The destination IP host will uses unicast MAC address as destination instead of broadcast MAC address because the sender MAC address is included in the ARP request received.

Now destination LEC sees this packet and finds that it has to go to a particular client. If it has MAC-to-ATM address details in his LE_ARP table, and the connection is available, then it sends the reply to him directly. Else it sends the ARP Response through the BUS itself. Now the sender host gets the ARP Response and IP-to-MAC relationship is established for the destination address.

Sender LEC sees the MAC address and looks up if it has it in its local LE_ARP table to see if MAC-to-ATM relationship details exist for this MAC address. If the relationship exists and a connection (Data Direct VCC) also exists, then it puts the packet to that VCC. Else it sends the packet to the BUS. The BUS sends the packet to all the clients so that the actual client picks it up while all others ignore the packet. If there is no Data Direct VCC available but the table has MAC-to-ATM relationship, then it starts the process of establishing the connection.

If the MAC-to-ATM relationship does not exist in the local sender LEC LE_ARP table, then it consults the LE Server by sending LE_ARP_REQUEST asking the ATM address for the destination MAC address. The LE Server will response to the LEC by sending it a LE_ARP_RESPONSE control frame that contains the ATM address for the destination. Then the sender LEC starts the process of establishing the connection.

3.1.5 Managing ATM Broadcast

Whenever a LEC receives a MAC data frame for transmission, the leading bit of the destination MAC address indicates whether the packet is unicast or broadcast/multicast:

- A leading 1 indicates a broadcast or multicast message.
- A zero indicates a unicast message.

Figure 3.3 illustrates an ATM broadcast interaction. The LEC immediately passes broadcast/multicast frames to the BUS via the VCC established initially (1). The BUS

forwards that message to all nodes on its point-to-multipoint tree—that is, all LECs on the emulated LAN receive the broadcast (2). If the BUS receives two broadcast or multicast frames simultaneously, it buffers one briefly while it sends the other one out. This serialization prevents intermixing of cells from different data frames on the VCC to the LECs. AAL5 allows a LEC to reassemble only one data frame at a time on a single VCC. Protocol information in the head of every frame uniquely identifies the LEC that originated the broadcast to the BUS. This information is the LEC ID, which the LES assigns. To speed broadcast frame processing, a LEC checks the ID for every incoming frame; if there is a match, the LEC disposes of the frame immediately. Otherwise it looks at the destination address. A non-proxy LEC saves only those frames whose destination MAC address matches its own multicast address. Proxy LECs save all multicast frames.

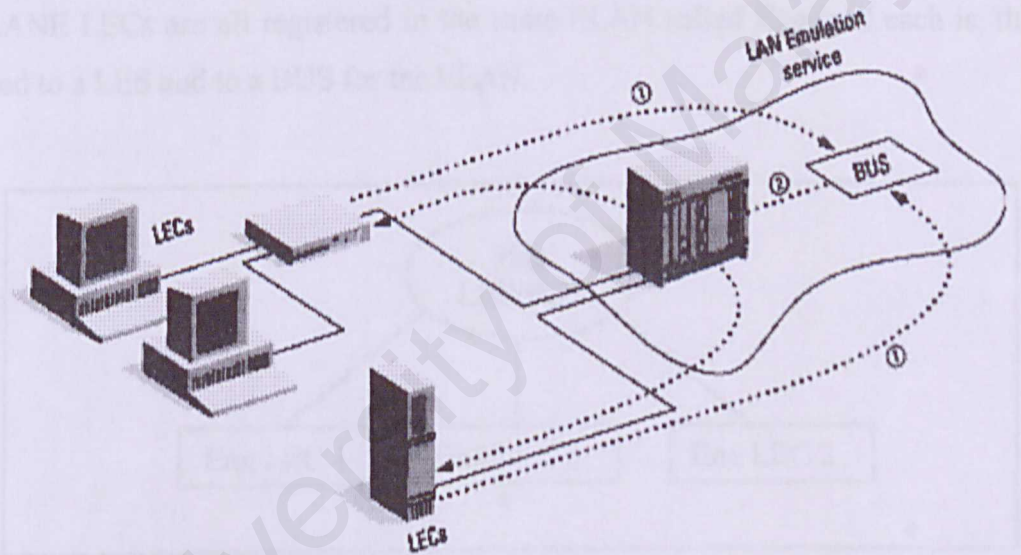


Figure 3.3: Broadcast and multicast message managed by the BUS

The BUS mechanism is designed to handle low-level broadcast traffic, such as IP address-resolution requests and SAP messages. However, it suffers from inefficiency when faced with vast quantities of broadcast frames. For example, suppose a LEC wanted to multicast live video to five LECs out of a population of 100 on an emulated LAN. Using the current distribution scheme, 95 LECs would end up discarding the transmitted frames immediately upon reception. Even high-capacity ATM networks can become congested when bandwidth is wasted. The BUS is allowed to set up point-to-point VCCs to the LECs, but the use of point-to-multipoint VCCs relieves the BUS from duplicating and transmitting many copies of each message.

3.1.6 Distributed LAN Emulation

Distributed LAN Emulation (DLE) Allows the LES and BUS functions that are provided to each ELAN to be distributed among multiple, interconnected server platforms. In this way, DLE provides these ELANs with resiliency and scalability.

To understand DLE operation, it is useful to compare DLE to the current service model, which uses a single LES and BUS for each ELAN. This section first describes a simple example of the single server model and then gives a detailed overview of the DLE model.

3.1.6.1 Single Server LANE Services Model

Figure 3.4 shows the topology of a single server supporting an ELAN. In this example, the LECs are hosts that are using IP, and the LES and BUS are running on the same switch. Three LANE LECs are all registered in the same ELAN called Eng, and each is, therefore, connected to a LES and to a BUS for the ELAN.

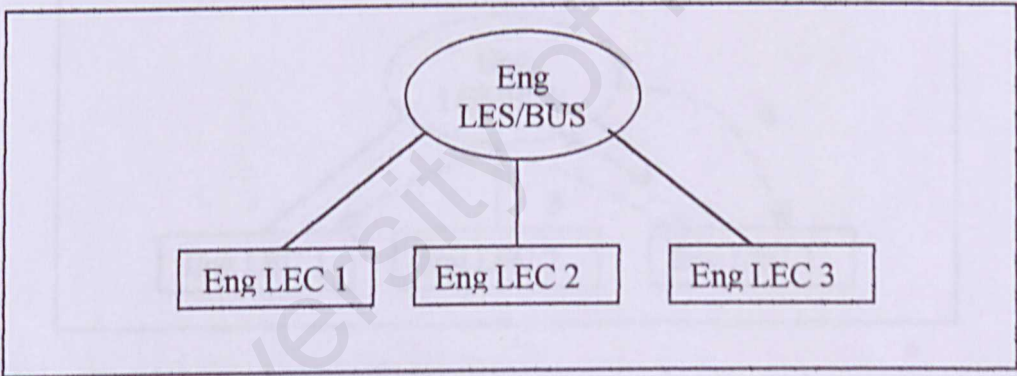


Figure 3.4: Single server LANE services model

3.1.6.1.1 Using a Single Server

When LEC 1 wants to connect to LEC 3, several messages are exchanged. First, LEC1 attempts to learn the MAC address of LEC 3 by requesting an IP-ARP request with LEC 3's IP address. As Figure 3.5 shows, this ARP request is sent in two steps: (1) as a point-to-point message from LEC 1 to the LANE BUS, then (2) as a point-to-multipoint message from the BUS to all of the LECs registered in the ELAN.

When LEC 3 receives the IP ARP request, it recognizes that it is the intended destination, and, therefore, attempts to send IP ARP response to LEC 1 (whose MAC address was supplied in the ARP request packet).

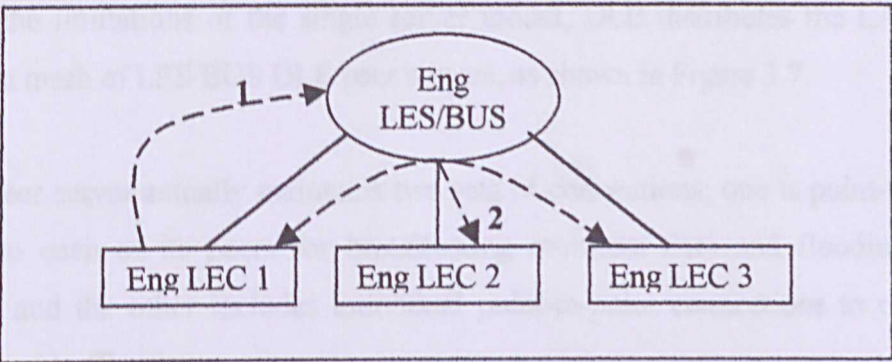


Figure 3.5: Broadcast IP-ARP request

As shown in Figure 3.6, the delivery of the ARP response is three step: (3) LEC 3 sends an LE-ARP query to the LES, asking for the ATM address corresponds to LEC 1’s MAC address; (4) the LES sends an LE-ARP response to LEC 3; and (5) LEC 3 establish a circuit to LEC 1’s ATM address.

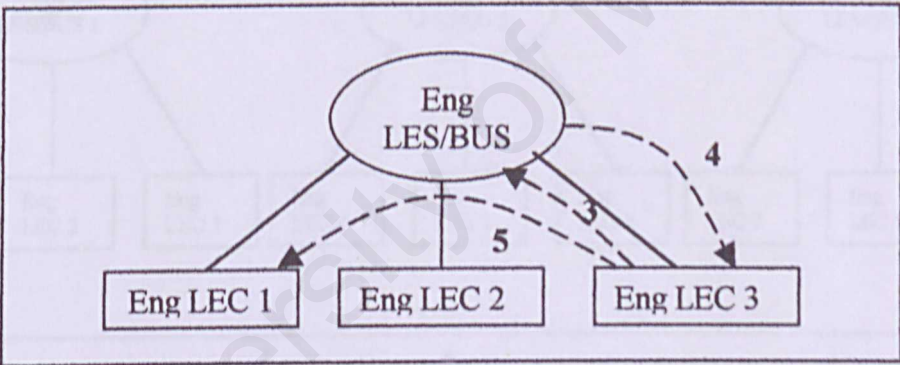


Figure 3.6: IP-ARP response handling

3.1.6.1.2 Limitations of a Single Server

Because there is only one LES/BUS supporting the ELAN, the following limitations exist:

- The number of LECs in a single ELAN is limited by the number if virtual circuits that the single LES/BUS can establish through their platform’s ATM port. This usually limits the ELAN to about 500 LECs.
- Cluster of LECs that are geographically separated from the LES/BUS may have poor throughput, even when connecting to each other, because address queries and broadcast may traverse slow wide-area links.
- A failure of the LES or BUS brings down the ELAN.

3.1.6.2 Distributed LAN Emulation Model

To address the limitations of the single server model, DLE distributes the LANE services load among a mesh of LES/BUS DLE peer servers, as shown in Figure 3.7.

Each DLE peer server actually maintains two sets of connections; one is point-to-multipoint connection to each of its peers for broadcasting multicast data and flooding control of information, and the other includes individual point-to-point connections to each peer for directed control traffic.

Each DLE peer server that supports the ELAN is responsible for registering and giving reports about the LECs that are attached to it directly. Each DLE peer server propagates this information to both its locally attached LECs and its peers.

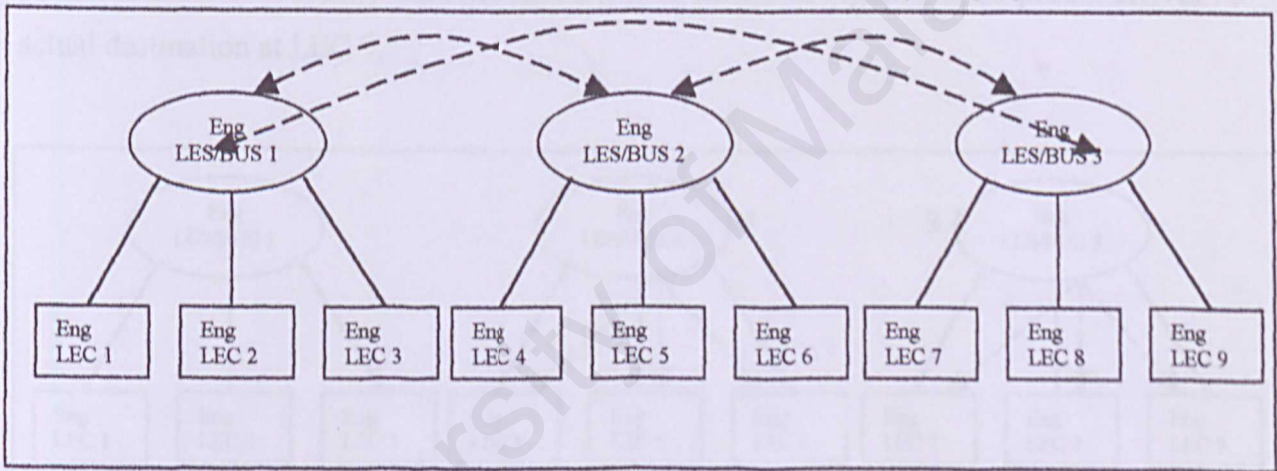


Figure 3.7: Distributed LAN emulation model

3.1.6.2.1 Using DLE

Figure 3.8 shows how a connection begins to be established through DLE peer servers. LEC 1 wants to communicate with LEC 9, which is in the same ELAN, but is locally attached to a different peer server. First, (1) LEC 1 sends an IP ARP broadcast request to its local DLE BUS. Then (2) the BUS broadcast the packet to both its locally LECs and its DLE peer servers.

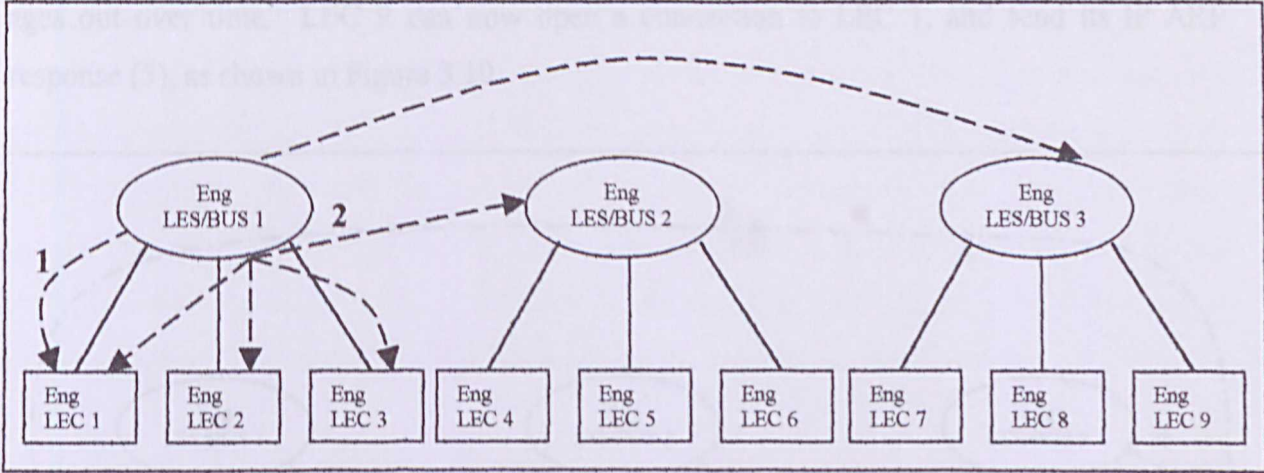


Figure 3.8: IP ARP broadcast from LEC 1 to LEC 9

Upon receiving the broadcast from the first DLE peer server, the peers re-distribute the packet to their own locally attached LECs (3), as shown in Figure 3.9, so packet arrives its actual destination at LEC 9.

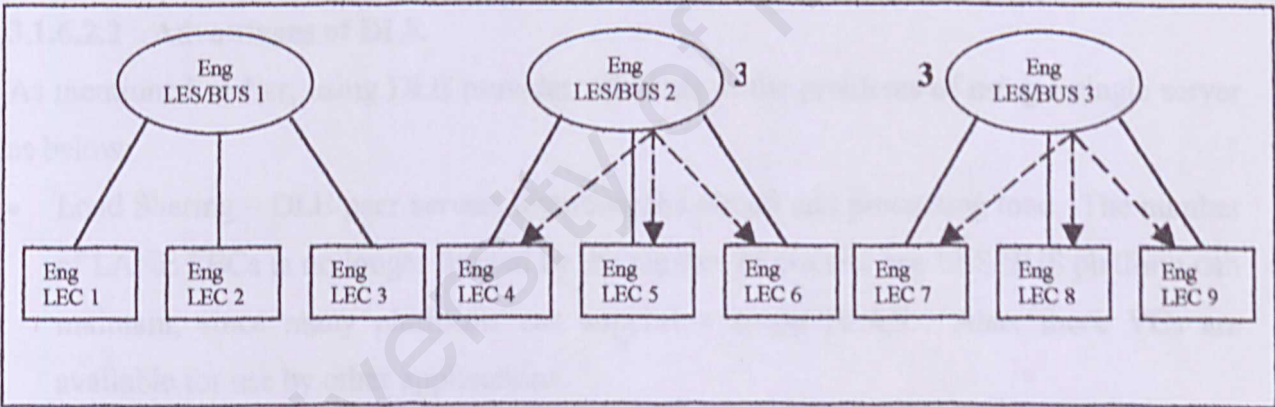


Figure 3.9: Re-distributing the broadcast across DLE peer servers

LEC 9 recognizes its IP address, and prepares an IP ARP response. LEC 9 then sends an LE-ARP request to its local LES, asking for the matches LEC 1’s MAC address. Since LEC 9’s local LES does not have an entry for LEC 1, the local LES passes the query along to all of its locally attached proxy LECs and all of its DLE peer servers. The first peer server has been able to resolve the LE-ARP response to the third server.

When the third DLE peer server receives the LE-ARP response, it passes it directly to LEC 9 (4). The third DLE peer server also caches the registration information for LEC 1 so that other local LECs do not have to go through the entire process again. However, this cache

ages out over time. LEC 9 can now open a connection to LEC 1, and send its IP ARP response (5), as shown in Figure 3.10.

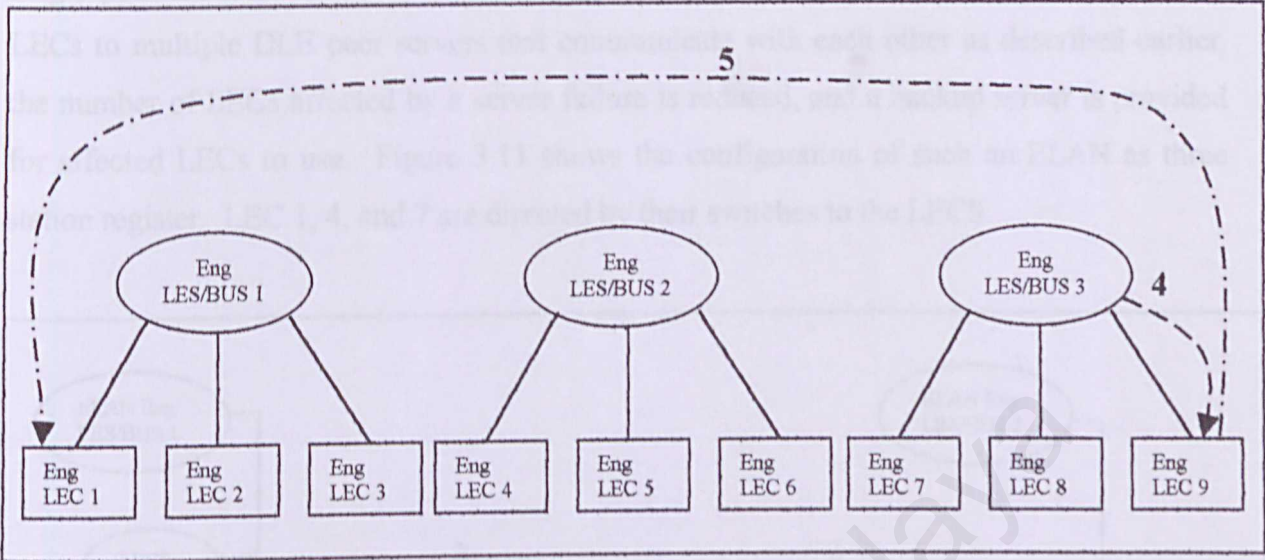


Figure 3.10: LE-ARP response delivered and LEC 9 contacts LEC 1

3.1.6.2.2 Advantages of DLE

As mentioned earlier, using DLE provides solutions to the problems of using a single server as below:

- Load Sharing – DLE peer servers distribute the circuit and processing load. The number of LANE LECs is no longer limited by the number of circuits one LES/BUS platform can maintain, since many platforms can support a single ELAN. Also, more VCs are available for use by other applications.
- Improved performance for remote LECs – With DLE, broadcast delivery and LE-ARP resolution across peer servers can take a little longer time than if all LECs were connected to a single server, since extra processing and transmission is needed. However, ELANs with groups of LECs indifferent locations can be designed for higher performance by providing a DLE peer server with each group. Broadcasts and address resolution within each group will be improved.
- Fault tolerance – The most advantage of DLE is fault tolerance. In a single serve ELAN, the server can be single point of failure. If the server fails, end stations in the ELAN are unable to discover each other through broadcast queries and unable to resolve MAC addresses into ATM addresses. In order to increased network reliability, therefore requires that ELANs have backups for LES/BUS functions.

3.2.6.3 DLE ELAN

A single server supporting an ELAN has a potential problem because the server can be a single point of failure. However, DLE can address this problem. By attaching the ELAN LECs to multiple DLE peer servers that communicate with each other as described earlier, the number of LECs affected by a server failure is reduced, and a backup server is provided for affected LECs to use. Figure 3.11 shows the configuration of such an ELAN as three station register. LEC 1, 4, and 7 are directed by their switches to the LECS.

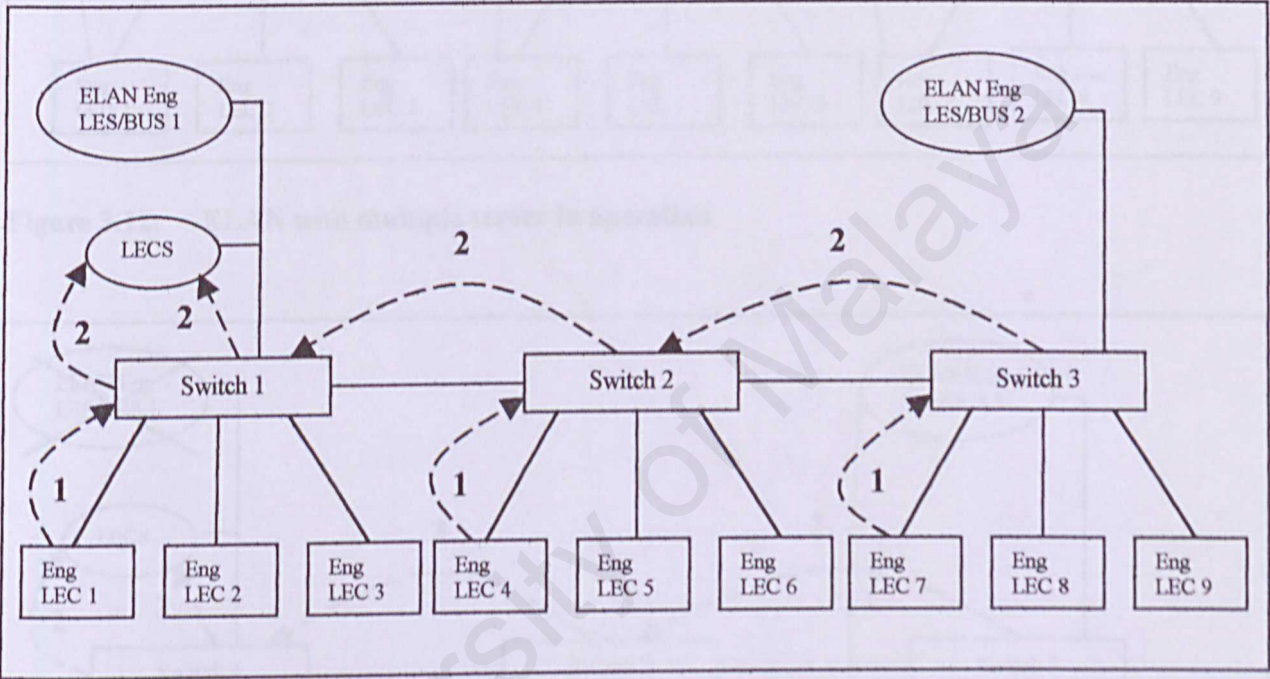


Figure 3.11: Registration on an ELAN with multiple servers

The LECS directed the LECs to register with the nearest LES/BUS by providing them the ATM address of LES/BUS. This is shown in figure 3.12.

This ELAN may experience significant performance improvements for the reasons described earlier. Even if the actual performance is similar to using single server in a particular network, a great advantage is gained through its fault tolerance if one of the server fails as depicted in Figure 3.13.

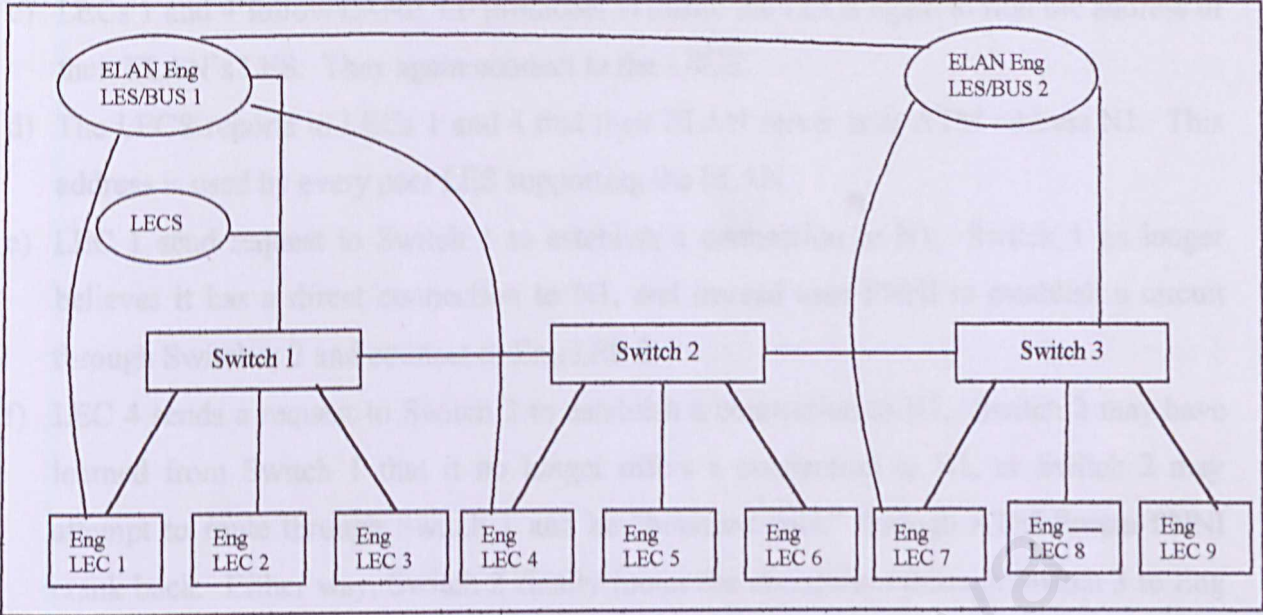


Figure 3.12: ELAN with multiple server in operation

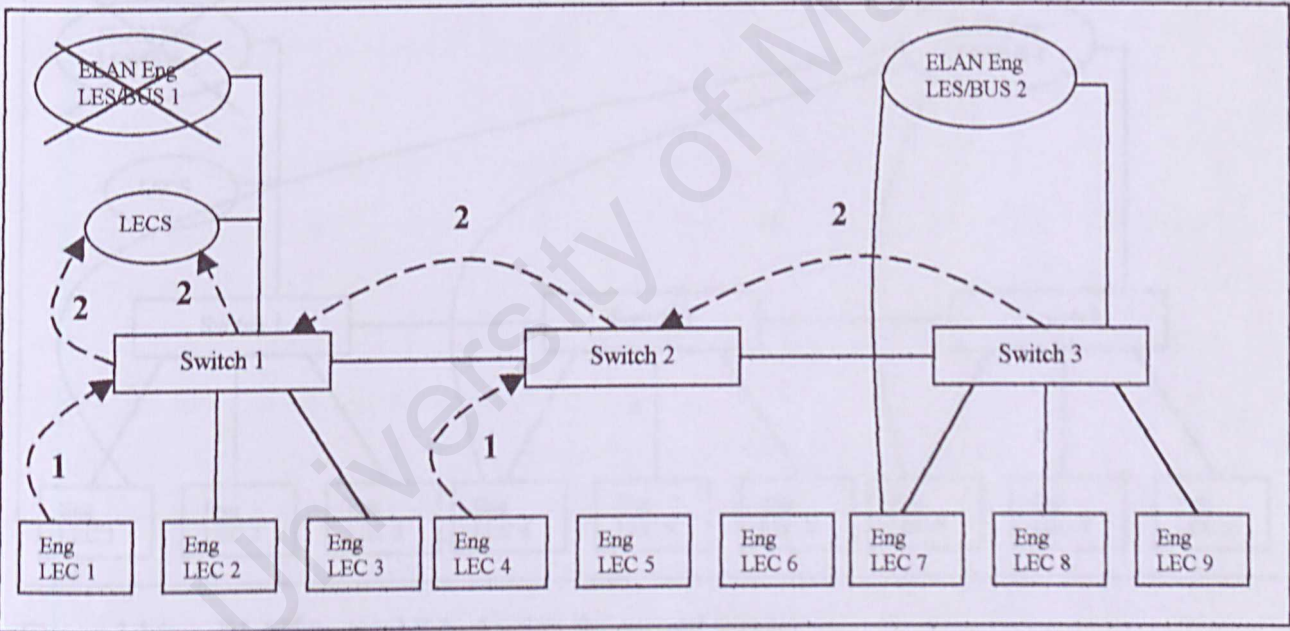


Figure 3.13: Failure of one ELAN server and the recovery process

The failure and recovery process occurs as follows:

- a) Eng LES/BUS 1 has lost power. All circuit connected to it are torn down. Low level signaling traffic stop, and Switch 1 removes the address of Eng LES/BUS 1 from its link tables.
- b) LECs 1 and 4 had been connected to switch 1. They detect that their connections to Eng LES/BUS 1 have been torn down; user intervention is not necessary.

- c) LECs 1 and 4 follow LANE 1.0 protocols to locate the LECS again to find the address of their ELAN's LES. They again connect to the LECS.
- d) The LECS reports to LECs 1 and 4 that their ELAN server is at ATM address N1. This address is used by every peer LES supporting the ELAN.
- e) LEC 1 send request to Switch 1 to establish a connection to N1. Switch 1 no longer believes it has a direct connection to N1, and instead uses PNNI to establish a circuit through Switches 2 and connect to Eng LES 2.
- f) LEC 4 sends a request to Switch 2 to establish a connection to N1. Switch 2 may have learned from Switch 1 that it no longer offers a connection to N1, or Switch 2 may attempt to route through Switch 1 and be "bounced back" through ATM Forum PNNI crank back. Either way, Switch 2 finally routes the connection through Switch 3 to Eng LES 2.

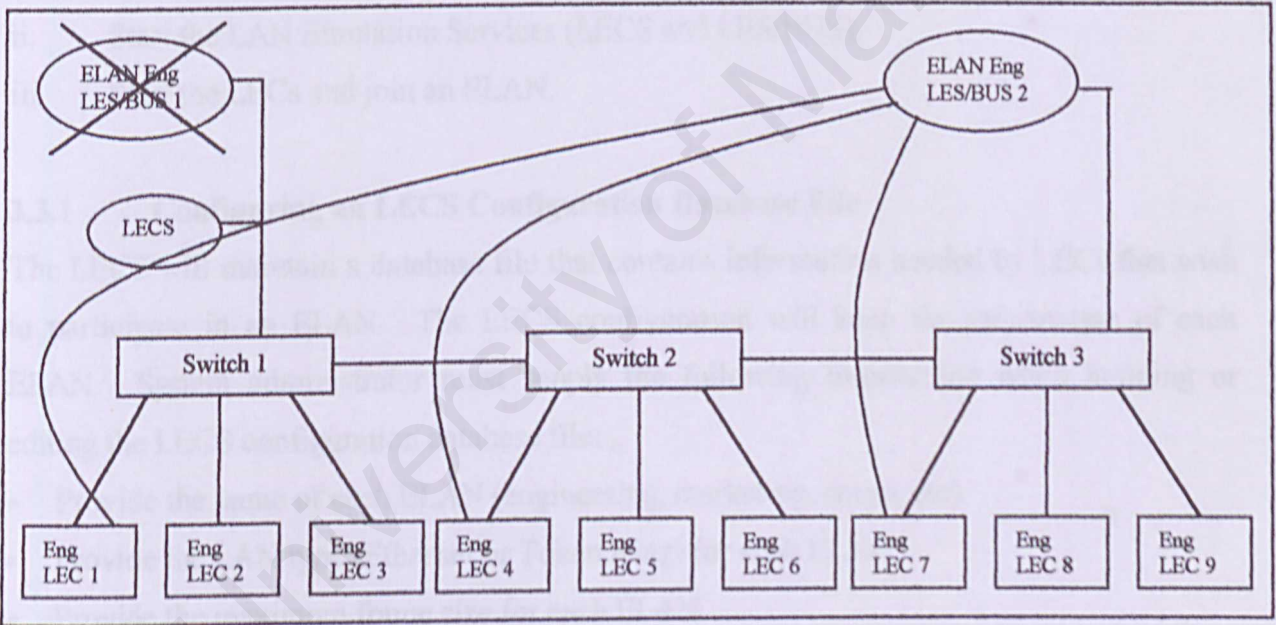


Figure 3.14: ELAN re-established using the second server

The recovery process occurs quickly – clients typically recover at the rate of 100 clients per minute – and the result is a reconfigured ELAN as shown in Figure 3.14.

3.2 ELAN Access Control

Basic ATM forum LAN Emulation Servers do not guard against unauthorized users learning an ELAN's LES address and then joining the ELAN. However, a method of authorization checking is available. After a LEC obtains the address of its LES, the LEC sends a request to

the LES to verify that the LEC is allowed to join. If the verification is received from the LECS, then the LES give the LEC permission to join. If verification is not received from the LECS, the LES rejects the join request and the LEC is dropped.

Using this feature, an authorization check is also performed each time the LECS reloads the LECS configuration file. The LECS periodically checks whether its configuration file has been modified, and, if it has, the file is re-read. If the file has changed to disallow some clients that were previously allowed, those clients will be dropped from the ELAN.

3.3 Configuring ELAN

There are three major steps that the system administrator should follow in order to configure and maintain ELANs:

- i. Configure an LECS configuration database file.
- ii. Start the LAN Emulation Services (LECS and LES/BUS).
- iii. Start the LECs and join an ELAN.

3.3.1 Configuring an LECS Configuration Database File

The LECS will maintain a database file that contains information needed by LECs that wish to participate in an ELAN. The LECS configuration will keep the information of each ELAN. System administrator must supply the following information when building or editing the LECS configuration database file:

- Provide the name of each ELAN (engineering, marketing, comp, etc).
- Provide the LAN type (Ethernet or Token Ring) for each ELAN.
- Provide the maximum frame size for each ELAN.
- Provide the ATM address for each ELAN. If DLE is supported in the ELAN, this address must be the anycast address for the DLE peer servers in each ELAN. Be sure to choose a distinct anycast address for each ELAN in the network. It must be unique within the first 19 bytes.
- Provide the address of each LEC that may participate in each ELAN
- If the LECs is able to use a default ELAN, the default LES information must also be included.

3.3.2 Defining an ELAN

The system administrator need to supply the information below when creating an ELAN:

- i. Provide the LES ATM address
- ii. Provide the name of the ELAN
- iii. Provide the maximum frame size of the ELAN
- iv. Provide the anycast ATM address and peer ATM address if DLE is supported

3.3.3 Starting the LECs and Joining an ELAN

After the ELAN services have been started, and LECs can join the ELAN. To start a LEC that will attempt to join the ELAN, the user need to supply the information below:

- i. Provide the LEC ATM address
- ii. Provide the ELAN name to join
- iii. Provide the IP address of LEC
- iv. Provide the netmask interface

3.4 Summary

The operation of LANE system and component include 4 stages of LEC operations, which is initialization and configuration, joining and registering with LES, finding and joining the BUS and data transfer. These operations in LANE are included to design the basic functionality of the network simulator.

CHAPTER 4

SIMULATION OVERVIEW AND DESIGN

CHAPTER 4: Simulation Overview and Design

This chapter covers the important features to be implemented into the simulator and the programming language and tool used to develop the simulator.

Besides that, it explains the design of the network simulator in details. The design aspects of the LANE network simulator components are focus in the design object classes since object-oriented approach is chosen to develop the simulator. The design of each component in the LANE network and the signaling uses in the LANE are explain in this chapter. It explains the functionality of each components and signals.

4.1 Overview of Simulation Model

The simulation features determine the functionality of simulator to replicates the real world behavior of the network and its ability to simulate the network. This section gives an overview of the features that are implemented into the simulator in order to provide accurate simulation output result.

4.1.1 Object Oriented Programming Approach

The simulator is built using object oriented programming approach. The OOP approach will be able to provide the capabilities of being reusable, flexible and extensible. In this simulation model, it consists of objects that used to simulate the behavior and states of real world objects.

This approach is used because it models the real world objects and thus reducing the complexity and the program structure is very clear. Besides that, it provide the benefits of modularity in which every object forms a separate entity whose internal workings are decoupled from other parts of the system. Besides that, it is easy to make minor changes in the data representation or the procedures in an OOP program. Changes inside a class do not affect any other part of a program, since the only public interface that the external world has to a class is through the use of methods. It able to add new features to the network simulator and this make the program extensible by introducing a few new objects and modifying some existing ones.

Thus, the benefits provided in object-oriented programming approach make it as the ideal approach to develop the network simulator. The use of this approach provides the network simulator with other features such as simplicity, modularity, extensibility, maintainability and reusability.

4.1.2 Multithreading

This simulator model makes use of parallel operation in which each object can run concurrently and simultaneously. Every component in the network is a single thread and all of them are responsible to determine when its own transaction will be performed. The use of multithreaded operation does not require constructing a global event list because each object handles its own event independently. A general clock is used to control the generation of next tick for each process to execute. This happens when not every process could complete their execution at the same time with other process. Therefore, they are designed to run concurrently and at the same time waiting for each other during execution time.

4.1.3 Portable

Another important feature of the network simulator is platform independent. Thus, this simulator can run on any platform and hardware without any recompilation or modification of the source code.

4.1.4 Programming Language Used

Java is chosen as the programming language used to develop the network simulator rather than C++. This is because it fulfills the vital features of the network simulator. Firstly, the most essential features of the network simulator are built in object-oriented approach. Java is a true object-oriented language. It does not merely provide the capability to implement object-oriented principles but it enforces these principles. Java provides all the luxuries of object-oriented programming: class hierarchy, inheritance, encapsulation, and polymorphism – in a context that is truly useful and efficient.

Secondly, multithreaded operation is another important feature of the network simulator. Java has a built-in support for multithreading and this allows parallel operation of objects within the network simulator.

Thirdly, the Java allows for the development of portable and web-based application. This enables the network simulator to run in any platforms without doing any modification on the program. Besides that, it the simulator can be run across the Internet by using Java compatible browser and increase the accessibility of the network simulator.

Apart from the core features, the simplicity, high performance, robustness features in Java has made it chosen as the programming language used to develop the network simulator. It provides all the necessary features of the network component.

Other object-oriented languages such as C++, DELPHI, Smalltalk is not chosen because they lack of providing the main features of the network simulator that are multithreaded operation, platform-independence and web-enabled ability.

4.1.5 Programming Tools

A programming tool is needed to support the uses of Java to develop the simulator. This tool must include the essential tools such as the Java language compiler, and additional tools such as debuggers, graphical user interface and Java language libraries.

Borland JBuilder 3.5 is chosen as the programming tools to develop the network simulator. JBuilder 3.5 is written completely in Java. It uses the Swing component library, provided by JavaSoft (Sun Microsystems), as its foundation. It provides a set of integrated tools that is easy to use in develop a platform independence and high performance application in Java.

JBuilder has a flexible architecture in which it is easy to incorporate the latest JDK version, third party tools, add-ins and Java Beans. The JBuilder IDE supports a variety of technologies including 100% Pure Java, JavaBeans, Java 2, Java SDK 1.2.2. and JFC/Swing.

JBuilder provides powerful and complete sets of tools including graphical user interface and editor have met the entire requirement for the development of this project.

4.2 JavaSim Network Simulator

This section will discuss on the design of Java Network Simulator developed by Master Student, Mr. Lim Shiau Hong. The LANE components are added into the simulator and it will be used to simulate the LANE topology.

4.2.1 Overview of JavaSim Network Simulator

The JavaSim object is the main object of the simulator. It keeps a list of all the network components, which are the descendents of SimComponent, and a list (a queue) of all events that in the form of SimEvent. Every component contains a set of parameters, which inherit SimParameter. All other classes are mostly helpers that provide certain services such as time service, logging and meter display. Figure 4.1 shows the hierarchy of all the significant objects in the simulator. All classes within the dotted rectangle belong to the simulation engine and should not be changed by component developers.

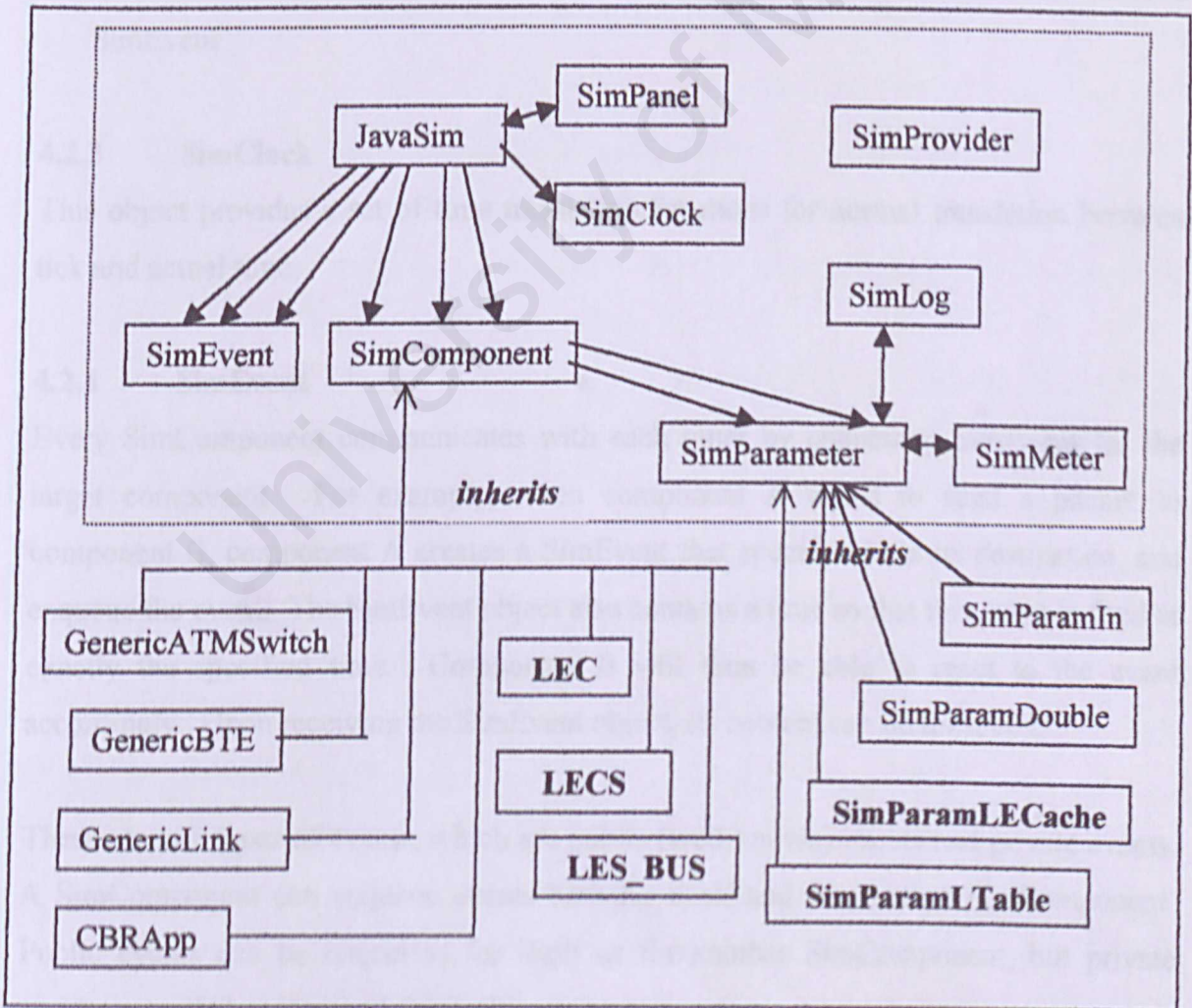


Figure 4.1: Hierarchy of Objects in the Simulator

The LANE components LECS, LES_BUS and LEC inherit SimComponent whereas new parameters used in the LANE components such as IP address, LECS's ELAN configuration database and LES registration table inherit SimParameter.

4.2.2 JavaSim

JavaSim is the main object that contains everything in the simulator. It provides all GUI functions (together with SimPanel) and main JFrame for the application. Besides that, it provides the event manager to handle event-passing among all components. There will be only one instances of the JavaSim object throughout the simulation.

There are few services provided by the JavaSim object as below:

- Provide the current simulation time in tick
- Provide a list of all existing SimComponent
- Provide communication between any components which involve creation of a SimEvent

4.2.3 SimClock

This object provides a set of time translation functions for normal translation between tick and actual time.

4.2.4 SimEvent

Every SimComponent communicates with each other by enqueueing SimEvent for the target component. For example, when component A wants to send a packet to component B, component A creates a SimEvent that specifies B as its destination, and enqueue the event. The SimEvent object also contains a time so that this event is fired at exactly the specified time. Component B will then be able to react to the event accordingly. Upon receiving the SimEvent object, its content can be retrieved.

There are two types of events, which are public (well-known) events and private events. A SimComponent can enqueue events both for itself and for another SimComponent. Public events can be enqueued for itself or for another SimComponent, but private events can only be enqueued for itself.

All the public events are defined in the SimProvider object. Any development of new SimComponent and event types require the recompilation of this object. All private events are defined within the particular SimComponent source itself. All private events must be greater than a constant (SimProvider.EV_PRIVATE) defined in SimProvider. So, the first private event will have a value of $\text{SimProvider.EV_PRIVATE} + 1$, the next $\text{SimProvider.EV_PRIVATE} + 2$, and so on.

4.2.5 SimComponent

This is the most important class to understand in the simulator in order to development new components. Every network component in the simulation inherits SimComponent.

The SimComponent class itself should not be instantiated because it only provides the skeleton for an actual component. A new component should extends SimComponent and override its various methods in order to provide meaningful operations for the component.

Below are the methods available in the SimComponent:

- *void start()* – perform any operation needed when the simulation starts
- *void reset()* – perform a reset operation in order to bring the status of the component back to the same status as if it is just newly created
- *void action(SimEvent e)* – this is the event handler of this component, and will be called by the simulator as the destination fires. Besides that, all private events will be handled in this method
- *Boolean isConnectable(SimComponent comp)* – this is called by the simulation engine when a component is about to be connected to this component. The *comp* is a reference to the new component.
- *void addNeighbor(SimComponent comp)* – this is called by the simulation engine when a new neighbor is connected to this component.
- *void removeNeighbor(SimComponent comp)* – this is called by the simulation engine when a neighbor is disconnected from this component.
- *void copy(SimComponent comp)* – this method is used to copy parameter values of another SimComponent of the same type. Normally the only parameters to copy are those that can be specified by the simulation user.

- *Image getImage()* – this method is used to load an image file to represent the component in the simulator.

Besides that, every *SimComponent* must have a component class and a component type, as defined in the *SimProvider* class. The *getCompClass()* method can be used to obtain the component class whereas *getCompType()* method can be used to get the component type.

4.2.6 SimParameter

Every *SimComponent* can have internal parameters or external parameters, which can be shown or accessible by users. All external parameters must inherit *SimParameter*. By extending *SimParameter*, one obtains parameter logging and meter display features automatically.

Any parameter that inherits *SimParameter* will provide a constructor that includes at least 4 parameters, which are name of the parameter, name of the component own the parameter, time when the parameter is created and whether the parameter can be logged in the log file.

The current simulation engine provides 3 general purpose classes that all inherit from *SimParameter*: *SimParamInt*, *SimParamDouble*, and *SimParamBool*. Obviously, these 3 objects provide support for integer, double and Boolean parameters. Extending *SimParameter* accordingly can create other types of parameters.

There is one important requirement to all parameters that may be added/removed after the creation of a component. Any addition or removal of *SimParameter* from the component's parameter list (*java.util.List params*) should be followed by a notification call to the main *JavaSim* object, by this statement: *theSim.notifyParametersChange(this)*, which ensure that any opened dialogs containing those parameters get updated/closed.

When a *SimParameter* is created with *isLoggable* true, it's value will not get logged when for example, a *setValue()* call is done. This is to avoid excessive or unnecessary logging of data. Each component is responsible in controlling the rate of logging. In

order to make sure a new value of a `SimParameter` is logged, one must call the `update(long tick)` method.

4.2.7 Object Serialization and Load/Save Function

The simulator uses object serialization as a form of light-weight persistence. This allows accurate saving and restoring of the simulation states without much effort from the components developers.

In this simulator, every `SimComponent` and `SimParameter` is serializable. This means that each `SimComponent` and `SimParameter` must implement the `java.io.Serializable` interface. This rule carries down to all class members of the particular `SimComponent` or `SimParameter`. That is, if a component contains a member that is not primitive java types, it should also be `Serializable`.

4.2.8 Create Route

In order to create a route from a source to destination, a user first needs to configure the routing table in the `GenericSwitch` object. The user has to specify the ATM address and output link in the routing table. This will enable the switch to put the arrival cell in the proper output port. Besides that, every switch, link and BTE will keep a call record table in order to pass the arrival cell to the desired destination.

4.3 Design of Components in the LANE

In this section, it explains the design of classes for each component. This includes their functionality, major attributes and major method in the class design.

4.3.1 LECS

LANE configuration Server is responsible to assign the address of LES of an ELAN upon the success of initialization of LECs to join an ELAN. The LECS maintain a table of LES address and ELAN name in order to reply in the `LE_CONFIGURE_RESPONSE`. It will assign the LECs according to the ELAN that they wish to join by providing the ATM address of the LES. Below are the functionalities of designed LECS:

- The LECs will use the information provided in the `LE_CONFIGURE_REQUEST` to generate an `LE_CONFIGURE_RESPONSE`. This response may indicate success

or failure, depending on whether the prospective LE Client is to be allowed to attempt to join an LE Server.

- LECS will validate the ATM address of LEC, LAN-type, and maximum frame size in the LE_CONFIGURE_REQUEST and decide whether the LEC is allowed to join the desired ELAN.

The algorithm used to validate the information in LE_CONFIGURE_REQUEST received from LEC is as below:

```

while there are ELAN info available{
    if ELAN name is unspecified or ELAN name equals ELAN name in ELAN info{
        if maximum frame size is unspecified or maximum frame size equals maximum frame
        size in ELAN info{
            if LAN type is unspecified or LAN type equals LAN type in ELAN info{
                if the LEC ATM address is valid to join the ELAN{
                    set STATUS = 0 to indicate success
                    response a successful LE_CONFIGURE_RESPONSE to the LEC
                }
            }
            else{
                set STATUS = 7 to indicate failure to join the ELAN due to security reason
                send a failure LE_CONFIGURE_RESPONSE to the LEC
            }
        }
    }
}

if the LEC is not allowed to join the ELAN due to insufficient information received{
    set STATUS = 1 to indicate failure to join the desired ELAN
    send a failure LE_CONFIGURE_RESPONSE to the LEC
}

```

- Upon the successful of the initialization, the LECS respond the LEC with a LE_CONFIGURE_SUCCESS that contain 0 (success) in the STATUS field. It fills in the LAN-TYPE, MAXIMUM-FRAME-SIZE, TARGET-ATM-ADDRESS and ELAN_NAME into the LE_CONFIGURE_RESPONSE control frame. IF the initialization failed, then it replies a LE_CONFIGURE_RESPONSE that does not contain 0 in the STATUS field.

The following table lists the design of the major attributes and methods of the LECS class.

Major Attributes	Major Methods
ATM address value LE_CONFIGURE_RESPONSE packet Table to keep LES and BUS information	Validate LE_CONFIGURE_REQUEST Set LE_CONFIGURE_RESPONSE success contents Set LE_CONFIGURE_RESPONSE failure contents Search for LES/BUS information

4.3.2 LES/BUS

Each ELAN needs at least 1 pair of LES/BUS to serve the LECs. In the simulator, the LES and BUS are located together and share an ATM Address. This is useful because both of them can share the LE_ARP table. The LES serve the LECs during the join phase and data transfer phase. Below are the functionalities of the designed LES:

- The LES will validate the LE_JOIN_REQUEST. The value for LAN type and maximum frame size must be compatible with that of the LES. It will check the MAC address and ATM address to prevent two LECs with the same MAC address or ATM address joins the ELAN.

Below is the algorithm used to validate the joining of LEC:

```
if ELAN Name not equals LES/BUS ELAN name{
    set STATUS=2
    send a failure LE_JOIN_RESPONSE control frame to LEC
}
else{
    if LAN type not equals LES/BUS LAN type{
        set STATUS=2
        send a failure LE_JOIN_RESPONSE control frame to LEC
    }
    else{
        if maximum frame size equals LES/BUS maximum frame size{
            while there are registered LEC info available in LE_Cache{
                if LEC ATM address equals ATM address in LE_Cache or LEC MAC
                address equal MAC address in LE_Cache{
                    set STATUS = 5 to indicate failure to join the ELAN due to duplicate
                    ATM address or MAC address
                    send a failure LE_JOIN_RESPONSE control frame to LEC
                }
            }
        }
        else{
            generate a LECID for the LEC
        }
    }
}
```

```

    add the new LEC info into LE_Cache
    set STATUS = 0
    send a success LE_JOIN_RESPONSE which included the LECID to the LEC
  }
}
else{
  set STATUS = 2 to indicate failure to join the ELAN
  send a failure LE_JOIN_RESPONSE control frame to LEC
}
}
```

- LES will return a LE_JOIN_RESPONSE to indicates a successful join and include a REQUESTER_LECID for the client that is unique among all LEC joined the same ELAN. It returns the same LE_JOIN_REQUEST it provided if the LEC did not received an LE_JOIN_RESPONSE.
- The LES respond to an LE_ARP_REQUEST for a registered LAN destination with the information obtained when that LAN destination was registered. If it does not respond to an LE_ARP_REQUEST, it must forward the LE_ARP_REQUEST. It can also forward the LE_ARP_REQUEST to the LEC that registered with that LAN. It can forward the LE_ARP_REQUEST to the peer server.

The following table lists the design of the major attributes and methods of the LES:

Major Attributes	Major Methods
ATM address	Validate LE_JOIN_REQUEST
ELAN name	Register the LEC
Maximum data frame	Sending LE_JOIN_RESPONSE
LE_Cache table to stored the information of joined LECs	Sending LE_UNREFISTER_RESPONSE
	Unregistered the LEC
	Responding to known LE_ARP_REQUEST
	Forwarding LE_ARP_REQUEST
	Sending LE ARP RESPONSE

A BUS is used to support the multicast function in the LANE. It receives the registration of LECs. Below are the major functionalities of the designed BUS:

- When BUS receives ARP_REQUEST from LEC to find the destination MAC address, it multicast the packet to all the LEC. It will also multicast the ARP_RESPONSE to the LECs.

- BUS is responsible to multicast the LE_ARP_REQUEST received from LES or LEC to find the destination ATM address.

The following table shows the major attributes and methods for the BUS:

Major Attributes	Major Methods
ATM address (same as LES)	Multicast the LE_ARP_REQUEST to LECs and peer server
	Multicast the ARP_REQUEST and ARP_RESPONSE received from LECs

4.3.3 LEC

An LEC act as a client in the ELAN and it can join more than one ELAN. The designed of LEC in this model are as below:

- During the initialization, the LEC issue a LE_CONFIGURE_REQUEST to the LECS containing LANE name that wish to join and its ATM and IP address. If the LE_CONFIGURE_RESPONSE does not contain 0, then the configuration phase has failed. If the LECS does not return an LE_CONFIGURE_RESPONSE within the LEC Control time-out, it can repeat the LE_CONFIGURE_REQUEST. If the LE_CONFIGURE_RESPONSE contain 0 in the STATUS field, then the configuration phase is successful.
- In the join phase, the LEC establishes its connection with the LES and issues a LE_JOIN_REQUEST to be allowed to join the ELAN. It receives a LE_JOIN_RESPONSE upon the successful of join phase.
- The LEC can send a LE_ARP_REQUEST to BUS for an unresolved unicast LAN destination. It can also send a LE_ARP_REQUEST to the LES to resolve that LAN destination.
- The LEC can send an ARP_REQUEST to the BUS to resolve a destination MAC address and reply to an ARP_REQUEST that asked for it MAC address by sending the ARP_RESPONSE to the BUS or directly through connection.
- The LEC must cache the LAN destination/ATM address mapping information received in at least those LE_ARP_RESPONSE corresponding to LE_ARP_REQUEST sent by the client.

- The LEC remove any LE_ARP cache entry that has not been used to forward a data frame for a long period of time, whether or no that entry has been recently verified.

The following table shows the major attributes and methods for the LEC:

Major Attributes	Major Methods
ATM address	Contact LECS
MAC address	Join ELAN
IP address	Send LE_ARP_REQUEST
Maximum retry count	Send ARP_REQUEST
Maximum frame size	Response to LE_ARP_REQUEST
ARP table	Response to ARP_REQUEST
LE_ARP table	Caching the ATM address
ELAN joined array	Established connection with other LEC
Aging time	Clear the LE_ARP cache
Destination IP address	
Destination ATM address	
Destination MAC address	

4.4 Additional Parameter Design

The parameter provided in the Java Network Simulator is insufficient to be used in the LANE component. Therefore, new parameters need to be created.

4.4.1 SimParamLTable

This parameter is specially designed to handle the ELAN database. It will enable user to input the details of each ELAN. It will require the users to input the name of the ELAN, ATM address, MAC address, LAN type, maximum frame size and display ELAN information in a table. It allow user to add and remove an ELAN from the database. Besides that, it performs checking function to make sure that the ATM address and MAC address are valid.

4.4.2 SimParamLECache

This parameter act as a cache table to stored the information of LECs that successfully joined the ELAN in LES/BUS. It performs the validation of LEC information, which does not allow two or more different LECs to register LAN destinations using the same ATM address or MAC address. It returns a unique LEC ID to the LEC that registered with the ELAN. Besides that, it can display the information of each LEC in a table.

4.4.3 SimParamMAC

SimParamMAC is specifically design for handling MAC addresses input by user. Basic functions include method to access all the parameters stored for instance read method and write method. Other method involve error checking method to ensure the correctness of parameter input by users

4.4.4 SimParamIP

This parameter is design to handle the IP address input by the user. It will validate the IP address input by the user to ensure the IP address is a valid type. It performs error checking on the IP address input by the user e.g. IP address must be numeric number between 0 to 255.

4.4.5 SimParamIntM

This parameter is used to limit the range of numeric that can be input by the user. It will perform a checking to make sure the value key in by the user is between a minimum value to a maximum value. For example, the value of maximum data frame size value is from 1516 octets to 18190 octets.

4.4.6 SimParamString

This parameter is design to handle string parameter key in by the user such as ELAN name. Besides that, it is used to show the status of each LANE component during the simulation.

4.5 User Interface Design

In the LANE simulator design, it consist of 5 sections as follow:

- LECS

In this section, user has to input LECS ATM address and ELAN configuration database, which will consist of the LES ATM address and the allowed NSAP to join the ELAN.

- LES/BUS

In this section, user needs to input the ELAN name, ATM address of LES/BUS, aging time and ELAN type. It will also consist a table that record the information of succeeded LECs joined the ELAN.

- LEC

In this section, it provide a few fields to be input by the user, which is LEC name, IP address, ATM address, MAC address, and the name of ELAN to join. In the data transfer section, the user only need to specify the destination IP address, number of bits to be sent. The user can change the destination address after the LEC finish sending the data.

- Log file

This section enables user to view the log file for LECS, LES/BUS, LEC, and the simulation result.

4.6 Summary

There are few important features to be implemented into the simulator such as using object-oriented approach, support of multithreaded, platform independence and web-enabled. Thus, Java programming language is chosen to develop the simulator because it meets the vital requirements of the network simulator.

This chapter covers the overview design for components of LANE to operate in the simulator. Each LANE components have their own functionality and task to be performed during the simulation. This will provides the major classes and functions to be included into the simulator.

CHAPTER 5

SIMULATOR IMPLEMENTATION

CHAPTER 5: Simulator Implementation

This chapter will cover the implementation aspect of the simulator components – a look into how the component is designed and implemented. During the implementation phase, all the classes with important attributes will be shown together with the explanation of these attributes as well as methods contained within the classes.

5.1 Implementation

The implementation of the ATM network simulator component is the phase that transforms the theoretical into the practical. It is where the actual simulator components are built from the foundation of the design that was put forth.

This section will look in turn at the implementation of each of the object classes that make up the LANE end system, as well as some of the other object classes that LANE end system makes use of.

5.1.1 Control Frame class

This class follows the LAN Emulation data frame format for IEEE 802.5 frames. It consists of the control frame use to carry the signaling information during the configuration phase, joining phase and data transfer phase of LEC.

```
class LaneFrame implements java.io.Serializable{
    String marker=null;
    int protocol=0;
    int version=0;
    int OP_CODE=0;
    int status=-1;
    int trans_id=0;
    int req_id=0;
    int flags=0;
    String src_lan_dest=null;
    String trg_lan_dest=null;
    String src_atm_addr=null;
    int src_port = 0;
    int resrv1=0;
    String trg_atm_addr=null;
    int trg_port=0;
    int resrv2=0;
    int lan_type=0;
    int max_frm_size=0;
    String elan_name="0"; }
```


5.1.2 LECS

This *LECS* class will perform the function of LANE configuration Server, which is responsible to assign the address of LES of an ELAN upon the success of initialization of LECs to join an ELAN. It will assign the LECs according to the ELAN that they wish to join by providing the ATM address of the LES.

During the configuration phase of LEC, LECS will receive the *LE_CONFIGURE_REQUEST*. The LECS will use the information from the configuration database to validate whether the LEC is allowed to join the intended ELAN. This is done by passing the *LE_CONFIGURE_REQUEST* to the *queryLES(Cell)* function which is available in the *SimParamLtable* class. If the initialization of LECs to join an ELAN is successful, it will call *LE_SUCCESS()* function to return a successful *LE_CONFIGURE_RESPONSE* to the LEC, otherwise it will call *LE_FAIL()* function which return a failure joining message to the LECs. The *LE_CONFIGURE_RESPONSE* is sent through the connection between LEC and LECS.

Below is the method in receiving *LE_CONFIGURE_REQUEST* from LECs during the configuration phase:

```
private void cn_receive(SimEvent e) {
    Cell cell=(Cell)(e.getParams())[0];
    if(cell.fr.OP_CODE==1)
    {
        LECS_STATUS.setValue("Get LE_CONF_REQUEST from" + cell.fr.src_atm_addr);
        LECS_STATUS.update(theSim.now());
        //validate the information of LE_CONFIGURE_REQUEST
        Cell rcell = ltable.queryLES(cell);

        if (rcell.fr.status!=0) //failure of configuration
            LE_FAIL(rcell, cell);
        else
            LE_SUCCESS(rcell, cell); //successful of configuration
    }
    else
        cell = null;
}
```

5.1.2.1 LECS GUI Implementation

There are seven attributes defined in the LECS class to hold the information such as ATM address, MAC address, start time, port number and LANE database. The figure below shows the GUI design of LECS component.

```
class LECS extends SimComponent implements java.io.Serializable {
```

```

    private SimParamNSAP nsap;           //LECS atm address
    private SimParamMAC mac;             // mac address
    private SimParamInt cn_start_time;   //start time
    private SimParamInt b_log_factor;    //logging factor

    private SimParamLTable ltable=null;  //table that keep the LANE information
    private SimParamInt cn_thisport=null; //port number
    private SimParamString LECS_STATUS;  //LECS status

```

```
}
```

5.1.3 LES/BUS

The LES_BUS class will perform the function of LES/BUS. It will receives LE_JOIN_REQUEST from LECs during joining phase and reply a LE_JOIN_RESPONSE to indicate whether the LEC has success or failed to join the ELAN.

For an LE_JOIN_REQUEST to succeed, values for LAN type and maximum frame size must be compatible with that LE server. The LES will include a REQUETER-LECID to indicate a successful join in the LE_JOIN_RESPONSE and the information of the LEC such as MAC address, ATM address, vpi and vci values of VCC will be kept in the *SimParamLECache*.

When LE server receive a LE_ARP_REQUEST from LECs, it will respond by checking the *SimParamLECache* to find the destination ATM address and return a LE_ARP_RESPONSE to the requested LECs. This task is performed by calling a *queryARP()* function in the *SimParamLECache*.

The BUS will able to response to ARP_REQUEST and ARP_RESPONSE. It will forward these request to all valid LECs. Besides that, it also transmits any valid data frame received from LECs.

5.1.3.1 LES/BUS GUI Implementation

The GUI parameter input window will be triggered when the LES component is created.

There are eleven attributes in LES component to hold the specific information as follows:

```
class LES_BUS extends SimComponent implements java.io.Serializable {
    private SimParamNSAP nsap;           //ATM address
    private SimParamMAC mac;             //MAC address

    private SimParamString ELAN_Name;    //ELAN name
    private SimParamString LAN_type;     //LAN type

    private SimParamInt b_log_factor;    //logging in ticks
    private SimParamIntM max_data_frame; //maximum data frame allowed
    private SimParamIntM aging_time;     //aging time to clear the cache
    private SimParamDouble cn_bit_rate;  //bit rate
    private SimParamInt cn_start_time;    //start time
    private SimParamLECache LE_Cache;    //table to store registered LECs information
    private SimParamInt cn_thisport=null; //port number
}
```

5.1.4 LEC

LEC class perform the functionality of a LEC in the LANE. It will go through the configuration phase, joining phase and data transfer phase. Below are the major methods in LEC class:

- void start()

This is the method where LEC performs configuration, register and transfer phase in the LANE simulation. In the implementation of LEC in the network simulator, it first needs to contact the LECS by sending a LE_CONFIGURE_REQUEST in order to get the ATM address of LES. It needs to set up a VCC to the LECS and then send the LE_CONFIGURE_REQUEST over the active connection. If the information in the LE_CONFIGURE_REQUEST not matches with the ELAN to join, then it will receive a failure of joining signal from the LECS. Otherwise, it will able to get the ATM address of the LES from the LE_CONFIGURE_RESPONSE received.

During joining phase, the LEC will first set up a VCC to contact the LES. The LEC will attached it own ATM address and MAC address inside the LE_JOIN_REQUEST packet. If the LEC received a join successful response, then it will get a LECID as its ID in the ELAN. Besides that, it will update LAN type, maximum data frame, ELAN Name and values given by the LE Server. This also indicate the joining phase of initialization is complete and

successful. If the LEC received a failure-joining message from the LES, it will retry the Join procedure. The number of retry must not exceed Maximum Retry Count.

During the data transfer phase, the LEC will first check the ARP table in order to map the IP address with MAC address. If the destination MAC address is not found, then it will send an ARP request to with broadcast MAC address as the destination address. LEC sees that and sends it through the Multicast send VCC to the BUS. If an LEC does not receive an ARP RESPONSE after control time out, it can retry to send the ARP request again. When a LEC received an ARP response from BUS, then it will update the information in the ARP table. Then the LEC will look up if it has it in its local LE_ARP table to see if MAC-to-ATM relationship details exist for this MAC address. If the relationship exists and a connection (Data Direct VCC) also exists, then it puts the packet to that VCC. Else LEC sends the packet to the BUS. The BUS sends the packet to all the clients so that the actual client picks it up while all others ignore the packet. If there is no Data Direct VCC available but the table has MAC-to-ATM relationship, then it starts the process of establishing the connection. Now if the MAC-to-ATM relationship does not exist in the local sender LE_ARP table, then it consults the LE Server by sending LE_ARP_REQUEST asking the ATM address for the destination MAC address. The LEC will receive a LE_ARP_RESPONSE from the BUS and update the information into the LE_ARP table. Then, it will set up a data direct VCC to the destination LEC and transmit data over the connection.

- *private void cn_receive(SimEvent e)*

In this method, it received LE_CONFIGURE_RESPONSE and LE_JOIN_RESPONSE during configuration phase and register phase. If LEC received an ARP request from BUS, and it finds that someone is asking for his MAC address, it coins ARP response and sends it down. If it has MAC-to-ATM address details in his LE_ARP table, and the connection is available (Data Direct VCC), then it sends the reply to him directly. Else it sends the ARP Response through the BUS itself. Besides that, it will be able to receive data from source LEC.

- *void DelCache()*

This function is built to check each entry in the LE_ARP table and delete an entry after the aging time is exceeded.

- *void findMAC()*

This method is used to retransmit an ARP_REQUEST to LES/BUS if the LEC does not received an ARP_RESPONSE after the time out.

- *Cell LE_CONF(Cell cell)*

This function is developed to attach the required information into the LE_CONFIGURE_REQUEST.

- *private LE_ARPinfo getLE_ARP(String targetmac)*

This method is used to map the MAC address to ATM address. It checked each entry in LE_ARP table to map with the MAC address of the destination LEC. If the mapping is successful, then it will return the target ATM address.

- *private ARP getMac(String cur_target_ip)*

This is called by the LEC to map IP address to MAC address. If the mapping is successful, it will return the target MAC address.

- *private Cell sentARP_request(String cur_targetIP)*

This is called by the LEC to send an ARP_REQUEST frame to BUS to resolve the IP address of the target LEC.

- *private Cell sentLE_ARP(BigInteger targetmac)*

This function is called to prepare an ARP_REQUEST control frame send to LES in order to resolve the ATM address of the target LEC.

The *ARP class* and *LE_ARPinfo class* are developed to store information in ARP table and LE_ARP table. Any new information of ARP response received will be record in the ARP table and LE_ARP_RESPONSE received will be updated into LE_ARP table. Besides that, the information in LE_ARP table will be clear after it reached the aging time. This is done by *DelCache()* function.

5.1.4.1 GUI Implementation of LEC

In the GUI implementation of LEC, the user will need to specify the ATM address, MAC address, IP address, ELAN Name, ELAN type, maximum frame time, maximum retry count, control time out, aging time and destination IP address. The user can change the destination IP address after the transfer of data is done in the previous connection. Below are the attributes in LEC component to hold the specific information:

```
class LEC extends SimComponent implements java.io.Serializable {
    private SimParamNSAP nsap;           //ATM address
    private SimParamMAC mac;             //MAC address
    private SimParamIP ip;               //IP address
    private SimParamIP subnet;           //Subnet mask

    private SimParamInt LEC_ID;           //LEC ID received from LES
    private SimParamInt b_log_factor;     //logging factor

    private SimParamString LAN_type;      //joined LAN type
    private SimParamIntM max_data_frame; //maximum data frame size
    private SimParamString ELAN_Name;     //joined ELAN name
    private SimParamIntM ctrl_time_out;   //control time out
    private SimParamIntM max_unknown_frame_cnt; //maximum unknown frame sent
    private SimParamInt max_unknown_frame_time; //maximum unknown frame time
    private SimParamIntM aging_time;      //aging time to clear entry in LE_ARP
    private SimParamIntM max_retry_cnt;   //maximum retry count
    private SimParamDouble cn_bit_rate;   //data transfer bit rate
    private SimParamInt cn_start_time;    //start time
    private SimParamDouble cn_trans_size; //transmission size
    private SimParamInt cn_delay;         //delay time to restart data transmission
    private SimParamInt cn_thisport=null; //port number

    private SimParamIP cn_destip;         //destination IP address
    private SimParamMAC cn_destmac;       //destination MAC address
    private SimParamNSAP cn_destnsap;     //destination ATM address
    private SimParamInt cn_conattempt;    //count of attempt connection
    private SimParamInt cn_conaccept;     //count of accepted connection
    private SimParamBool isRegLECS;       //check if register with LECS
    private SimParamBool isRegLES;       //check if register with LES/BUS
    private SimParamString LEC_STATUS;    //status of LEC
    private SimParamString LEC_STATUS2;  //data received
}
```

5.1.5 SimParamLTable

This class will act as the LANE database, which allowed the user to input the information of the LANE available. In this LANE database, the user needs to specify the name of the ELAN, its ATM address, ELAN types, maximum frame size, and the prefix of the ATM

address of LECs that is allowed to join the ELAN. It performs the function to validate the LE_CONFIGURE_REQUEST received from LECs.

```

class SimParamLTable extends SimParameter implements ActionListener, java.io.Serializable {
    private class LInfo implements java.io.Serializable{
        BigInteger nsap;           //LES/BUS ATM address
        BigInteger Ansap;         //Allowed NSAP address
        int mask;                  //mask
        int max_frame_size;        //maximum frame size
        String LANE_Name;          //ELAN name
        String LANE_Type;          //LAN type
    }
    private SimComponent theComp;
    private java.util.List itable;           //list to store ELAN information
    private LInfo info;
    private transient JComponent jcomp=null;
    String[] LT = {"Ethernet", "TokenRing"};
    String[] max_fr = {"1516", "4544", "9234", "18190"};
}

```

5.1.6 SimParamLECache

SimParamLECache class is built to keep the information of LEC that has successfully joined the ELAN. It performs the function to validate the LE_JOIN_REQUEST in order to prevent two LEC with same MAC address or ATM address join the ELAN. It will also return a LECID to LES if the LEC is qualified to join the ELAN. **SimParamLECache** will display the information of the LEC in a table.

```

class SimParamLECache extends SimParameter implements ActionListener, java.io.Serializable {
    class Info implements java.io.Serializable{
        BigInteger nsap;           //LEC nsap address
        BigInteger mac;           //LEC MAC address
        int lecid;                 //LEC ID
        int vci;                   //VCI value
        int vpi;                   //vpi value
        long c_time;              //register time
    }
    private SimComponent theComp;
    private java.util.List itable;           //list to store registered LEC information
    private Info info;
    private transient JComponent jcomp=null;
    private int lecid=1;
}

```

5.2 Conclusion

This chapter presented an idea on the implementation of LANE components. Class implementation explains the attributes in each component together with their data type. The major methods and attributes in each class are explained in the implementation.

CHAPTER 6. Testing

Chapter 6 discusses the testing process that must be followed for any software. This is the testing to determine whether your development testing will not find needed errors. The majority of the errors found in the simulation and product testing phases are the responsibility of failure management. Finally, developing software will not be under control as a whole.

CHAPTER 6

TESTING

University of Malaysia

	Value entered by the user	Expected value of the user	Expected
WWW address	100	100	100
WWW address	100	100	100
IP address	10.10.10.1	10.10.10.1	10.10.10.1
Mail address	100	100	100
WWW address	100	100	100
IP address	10.10.10.1	10.10.10.1	10.10.10.1
Mail address	100	100	100
WWW address	100	100	100
IP address	10.10.10.1	10.10.10.1	10.10.10.1

CHAPTER 6: Testing

Chapter 6 discusses the testing phases that need to be done for the simulator. Simulator testing is done in three parts. Component testing will test the on the correct functionality of the GUI input to the simulator and module testing focuses on the functionality of LANE components. Finally, the system testing will test the entire system as a whole.

6.1 Component Testing

The component testing is performed to verify that the values for each variables defined are correctly captured and transferred. Additional debugging codes are added into the component classes to verify the values entered by the user.

6.1.1 LANE Component Testing

The test is performed to check the correct value of ATM address and MAC address entered into LECS, LES/BUS and LEC component. Besides that, the IP address entered into LEC is also performed. The *System.out.println* statement is used for testing purpose:

```
System.out.println("NSAP address: " + nsap.getValue());
System.out.println("MAC address: " + mac.getValue());
System.out.println("IP address: " + IP.getValue());
```

6.1.1.2 Testing Result

The tests are executed using several different ATM address and MAC address. The correct values entered are expected for the output from the additional debugging codes.

Table 6.1: LANE Component testing result

	Values entered by the user	Expected output	Output
1. NSAP address	1301	1301	1301
MAC address	3301	3301	3301
IP address	10.10.1.1	10.10.1.1	10.10.1.1
2. NSAP address	ff01	ff01	ff01
MAC address	1e02	1e02	1e02
IP address	202.185.109.1	202.185.109.1	202.185.109.1
3. NSAP address	1a2b	1a2b	1a2b
MAC address	312e	312e	312e
IP address	201.175.139.2	201.175.139.2	201.175.139.2

6.1.2 SimParamLTable Component Testing

This test is to check the correct values for the parameter entered by the user into the SimParamLTable component and the consistency of the values passed from the component to the switch component. The coding bellow are used for the testing purpose:

```
System.out.println("ELAN Name: " + LANE_NAME);
System.out.println("LES NSAP address: " + nsap);
System.out.println("Allowed NSAP: " +Ansap);
System.out.println("Mask: " + mask);
System.out.println("Max Frame Size: " + max_fr_size);
System.out.println("LAN type: " + lan_type);

System.out.println("ELAN Name passed: " + rcell.fr.elan_name);
System.out.println("LES NSAP address passed: " + rcell.fr.trg_atm_address);
System.out.println("Max Frame Size passed: " + rcell.fr.max_frm_size);
System.out.println("LAN type passed: " + rcell.fr.lan_type);
```

6.1.2.1 Testing Result

The tests are executed using several different ELAN name, ATM address, allowed ATM address, mask, maximum frame size and LAN type. The correct values are expected for the output from the debugging statement.

		Values entered by the user	Expected output	Output
1.	ELAN name	Default	Default	Default
	LES NSAP address	1302	1302	1302
	Allowed NSAP address	0	0	0
	Mask	0	0	0
	Max frame size	1516	1516	1516
	LAN type	Ethernet	Ethernet	Ethernet
2.	ELAN name	Eng	Eng	Eng
	LES NSAP address	1402	1402	1402
	Allowed NSAP address	1400	1400	1400
	Mask	152	152	152
	Max Frame size	1516	1516	1516
	LAN type	Ethernet	Ethernet	Ethernet

3.	ELAN name	Comp	Comp	Comp
	LES NSAP address	1504	1504	1504
	Allowed NSAP address	1501	1501	1501
	Mask	160	160	160
	Max Frame size	1516	1516	1516
	LAN type	Ethernet	Ethernet	Ethernet

		Value passed by LECS	Expected output	Output
1.	LES NSAP address	1304	1304	1304
	ELAN Name	Default	Default	Default
	Max Frame Size	1516	1516	1516
	LAN Type	Ethernet	Ethernet	Ethernet
2.	LES NSAP address	1402	1402	1402
	ELAN Name	Eng	Eng	Eng
	Max Frame Size	1516	1516	1516
	LAN Type	Ethernet	Ethernet	Ethernet
3.	LES NSAP address	1504	1504	1504
	ELAN Name	Comp	Comp	Comp
	Max Frame Size	1516	1516	1516
	LAN Type	Ethernet	Ethernet	Ethernet

Table 6.2: SimParamLtable testing result

6.1.3 SimParamLECache Component Testing

This test is to check the correct values for the parameter add into the LE_Cache of LES/BUS. The information of LECs that successfully joined the ELAN will add into this table. Besides that, it checks the consistency of values passed from the LES/BUS to the switch component.

The coding show below are used for the testing purpose:

```
System.out.println("LEC ID: " + lec_id);
System.out.println("NSAP address: " + nsap);
System.out.println("MAC address: " + mac);

System.out.println("VPI: " + cell.vpi);
System.out.println("VCI: " + cell.vci);
System.out.println("NSAP address passed: " + cell.fr.trg_atm_address );
System.out.println("MAC address passed: " + cell.fr.trg_lan_dest);
System.out.println("LEC ID passed: " + cell.fr.req_id );
```


6.1.3.1 Testing Result

The tests are executed using several different LEC ID, ATM address and MAC address. The Table 6.3 shows the testing result of the component.

		Value entered by user	Expected value	Output
1.	LEC ID	1	1	1
	NSAP address	1401	1401	1401
	MAC address	401	401	401
2.	LEC ID	2	2	2
	NSAP address	1402	1402	1402
	MAC address	402	402	402
3.	LEC ID	3	3	3
	NSAP address	1501	1501	1501
	MAC address	501	501	501

		Value passed by LES/BUS	Expected value	Output
1.	LEC ID	1	1	1
	NSAP address	1401	1401	1401
	MAC address	401	401	401
2.	LEC ID	2	2	2
	NSAP address	1402	1402	1402
	MAC address	402	402	402
3.	LEC ID	3	3	3
	NSAP address	1501	1501	1501
	MAC address	501	501	501

Table 6.3: SimParamLECache testing result

6.2 Module Testing

The module testing is performed to check the functionalities of LECS, LES/BUS and LEC. The LECS is tested to verify the LE_CONFIGURE_REQUEST packet received from LECs during the configuration phase whereas LES/BUS is tested to validate the register packet from LECs on registration phase. LEC

6.2.1 Configuration Function Testing

The LE_CONFIGURE_REQUEST is verified by the LECS to ensure that the LEC is able to join the desired ELAN. It checked the information in the packet and decided whether the LEC is allowed to join the ELAN. Below is the configuration of the ELAN database:

ELAN Attributes		ELAN information
1.	ELAN Name	Default
	NSAP address	1304
	Allowed NSAP address	0
	Mask	0
	Max frame size	1516
	LAN Type	Ethernet
2.	ELAN Name	Comp
	NSAP address	1402
	Allowed NSAP address	1400
	Mask	152
	Max frame size	1516
	LAN Type	Ethernet

Table 6.4: LANE Database Configuration

6.2.2.1 Testing Result

The test is executed by adding a few LEC to join different ELAN. Table 6.5 shows the configuration packet sent to LECS and the result of configuration testing.

		Value input by user
1.	NSAP address	1403
	MAC address	403
	ELAN Name	Default
	Max frame size	1516
	LAN type	Ethernet
	Expected Result	Configuration successful
	Result	Configuration successful
2.	NSAP address	1502
	MAC address	502
	ELAN Name	0
	Max frame size	1516
	LAN type	Ethernet
	Expected Result	Configuration successful
	Result	Configuration successful
3.	NSAP address	1303
	MAC address	303
	ELAN Name	Comp
	Max frame size	1516
	LAN type	Ethernet
	Expected Result	Configuration failure – NSAP address not match
	Result	Configuration failure
4.	NSAP address	1405
	MAC address	405
	ELAN Name	Comp
	Max frame size	1516
	LAN type	Ethernet
	Expected Result	Configuration successful
	Result	Configuration successful

Table 6.5: Configuration function testing result

6.2.2 Registration Function Testing

This test is performed to verify the LE_JOIN_REQUEST received from LECs during registration phase. The LES/BUS will not allowed two LECs with same ATM address or MAC address to join the ELAN. Below is the configuration of the LES/BUS:

	Value input by user
1. ELAN Name	Default
NSAP address	1404
MAC address	404
Max frame size	1516
LAN Type	Ethernet

Table 6.6: Configuration of LES/BUS

6.2.2.1 Testing Result

The tests are executed using several LECs to join the ELAN. The LES/BUS able to decide whether the LEC is allowed to join the ELAN.

	Value input by user
1. NSAP address	1403
MAC address	403
ELAN Name	Default
Max frame size	1516
LAN type	Ethernet
Expected Result	Registration successful
Result	Registration successful
2. NSAP address	1502
MAC address	502
ELAN Name	0
Max frame size	1516
LAN type	Ethernet
Expected Result	Registration successful
Result	Registration successful

3.	NSAP address	1303
	MAC address	303
	ELAN Name	Comp
	Max frame size	1516
	LAN type	Ethernet
	Expected Result	Registration failure – ELAN Name not match
	Result	Registration failure
4.	NSAP address	1405
	MAC address	405
	ELAN Name	Default
	Max frame size	4544
	LAN type	Ethernet
	Expected Result	Registration failure – Max frame size not match
	Result	Registration failure

Table 6.7: Registration function testing result

6.2.3 Data Transfer Function Testing

The purpose of this testing is to transfer packets of data from a source LEC to a destination LEC. Besides that, it is also ensure that the source LEC will follows the process of data transfer phase and destination LEC received the cells in sequence.

6.2.3.1 Testing Result

		Source LEC	DestinationLEC
1.	ATM address	1404	1406
	MAC address	404	406
	IP address	202.185.1.2	202.185.1.3
	Subnet Mask	255.255.255.0	255.255.255.0
	ELAN Joined	Default	Default
	Expected Result	Data transfer successfully	Data received in sequence
	Result	Data transfer successfully	Data received in sequence

2.	ATM address	1306	1502
	MAC address	502	502
	IP address	202.185.2.3	202.185.2.4
	Subnet Mask	255.255.255.0	255.255.255.0
	ELAN Joined	Comp	Comp
	Expected Result	Data transfer successfully	Data received in sequence
	Result	Data transfer successfully	Data received in sequence
3.	ATM address	1404	1306
	MAC address	404	306
	IP address	202.185.1.2	202.185.2.3
	Subnet Mask	255.255.255.0	255.255.255.0
	ELAN Joined	Default	comp
	Expected Result	Data transfer failure – cannot find destination LEC	No data received
	Result	Data transfer failure – cannot find destination LEC	No data received

Table 6.8: Data transfer function testing result

6.3 System Testing

A topology as shown in Figure 6.1 is built to test the system. In this topology, there are two switches, one LE Configuration Server, two LE Servers named default and comp, and seven LE Clients. The prefix of ATM address under Switch 1 is 1300 whereas the prefix of ATM address under Switch 2 is 1400. The IP address of LECs, which join default ELAN will range from 202.185.2.2 to 202.185.2.4 and the IP address of LECs that join comp ELAN will range from 202.185.3.2 to 202.185.3.5. In the configuration database of LECS, any LEC is allowed to join default ELAN but only LECs with prefix of ATM address 1400 are allowed to join comp ELAN.

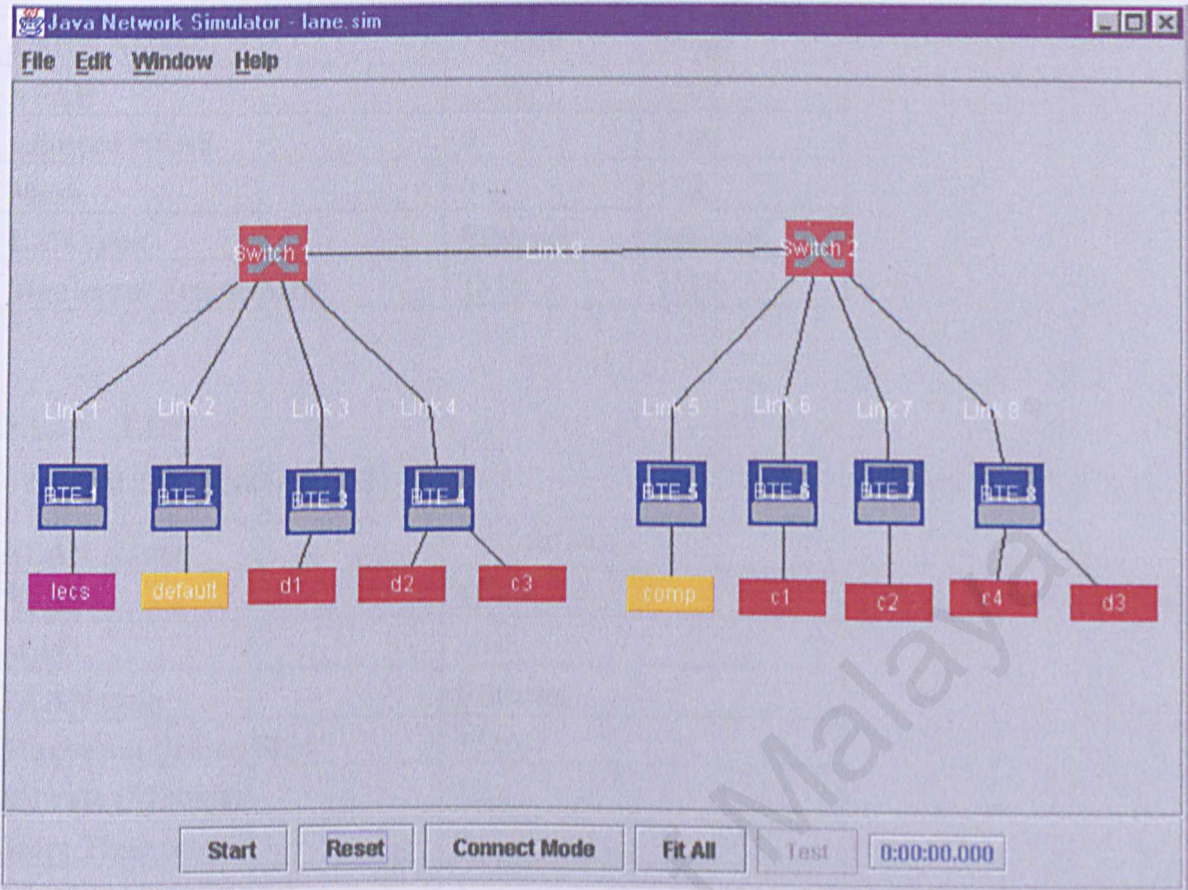


Figure 6.1: Simulation Topology

6.3.1 Parameter Configuration of LANE Component

This section details the configuration of each LANE component in the topology. The configuration of the components will affect the result of simulation and below are configuration of each component in the topology.

6.3.1.1 LE Configuration Server

The parameters of LECS are set as below:

NSAP(hex)	1302
MAC(hex)	302
Start Time (usec)	0
Logging every ticks (usec)	1

The LANE database is configured as bellow:

LANE Name	Default	comp
NSAP	1304	1402
Allowed NSAP	0	1400
Mask	0	152
LAN type	Ethernet	Ethernet
Maximum Frame Size	1516	1516

6.3.1.2 LES

- LES/BUS of default ELAN

ELAN Name	default
NSAP	1304
MAC	304
ELAN type	Ethernet
Maximum Frame Size	1516
Bit rate (Mbits/s)	10
Start Time (usecs)	0
Logging every (ticks)	1

- LES/BUS of comp ELAN

ELAN Name	comp
NSAP	1304
MAC	304
ELAN type	Ethernet
Maximum Frame Size	1516
Bit rate (Mbits/s)	10
Start Time (usecs)	0
Logging every (ticks)	1

6.3.1.3 LECs

In the simulation, LEC d1, d2 and d3 will join default ELAN whereas LEC c1, c2, c3, c4 will join comp ELAN. The parameters of LECs are show as below:

- LECs that join default ELAN

LEC Name	d1	d2	d3
NSAP	1306	1308	1409
MAC	306	308	409
IP address	202.185.2.2	202.185.2.3	202.185.2.4
Subnet mask	255.255.255.0	255.255.255.0	255.255.255.0
Logging every (ticks)	1	1	1
Maximum data frame	1516	1516	1516
Control time out (sec)	1	1	1
Max unknown frame count	1	1	1
Max unknown frame time (usecs)	1000000	1000000	1000000
Max retry count	1	1	1
Aging time (sec)	5	5	5
Start time (usecs)	0	0	0
Bit Rate (Mbits/s)	15	10	5
Number of Mbits to be sent	15	15	15
Delay between calls	1000000	1000000	1000000
ELAN to join	default	default	default
ELAN type	Ethernet	Ethernet	Ethernet
Destination IP address	202.185.2.4	202.185.2.2	202.185.2.2

- LECs that join comp ELAN

LEC Name	c1	c2	c3	c4
NSAP	1404	1406	1309	1408
MAC	404	406	309	406
IP address	202.185.3.2	202.185.3.3	202.185.3.4	202.185.3.4
Subnet mask	255.255.255.0	255.255.255.0	255.255.255.0	255.255.255.0
Maximum data frame	1516	1516	1516	1516
Logging every (ticks)	1	1	1	1
Control time out (sec)	1	1	1	1
Max unknown frame count	1	1	1	1
Max unknown frame time (usecs)	1000000	1000000	1000000	1000000

Max retry count	1	1	1	1
Aging time (sec)	6	5	5	5
Start time (usecs)	0	0	0	100
Bit Rate (Mbits/s)	15	10	5	5
Number of Mbits to be sent	15	15	15	15
Delay between calls	1000000	1000000	1000000	1000000
ELAN to join	comp	comp	comp	comp
ELAN type	Ethernet	Ethernet	Ethernet	Ethernet
Destination IP address	202.185.3.3	202.185.3.2	202.185.3.5	202.185.3.4

6.3.2 Simulation Testing

The testing of the simulation is divided into three phases, which are configuration phase, joining phase and data transfer phase.

6.3.2.1 Configuration Phase

In this phase, all LECs will first contact with LECS by sending a LE_CONFIGURE_REQUEST packet to find the location of LES. LECS will return an LE_CONFIGURE_RESPONSE packet to indicate whether they are allowed to join the ELAN.

Expected Result

In this phase, all LECs will be able to get the ATM address of LES except LEC c3. This is because its ATM address does not allow it to join comp ELAN.

Simulation Result

LEC c3 is not allowed to join comp ELAN whereas other LECs receive an LE_CONFIGURE_RESPONSE that includes the ATM address of the LES to join. Those LECs will proceed to the joining phase to contact the LES and BUS.

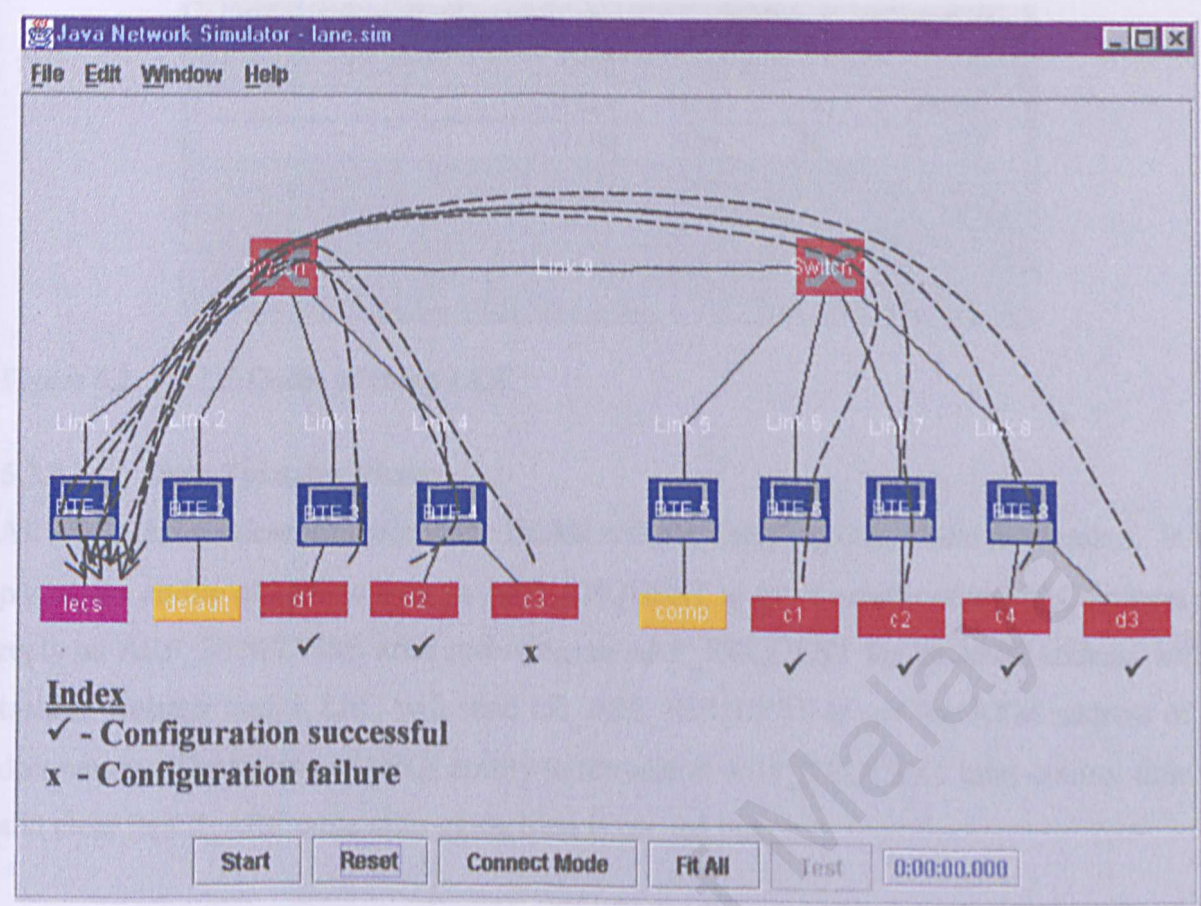


Figure 6.2: Simulation result of configuration phase

6.3.2.2 Join Phase

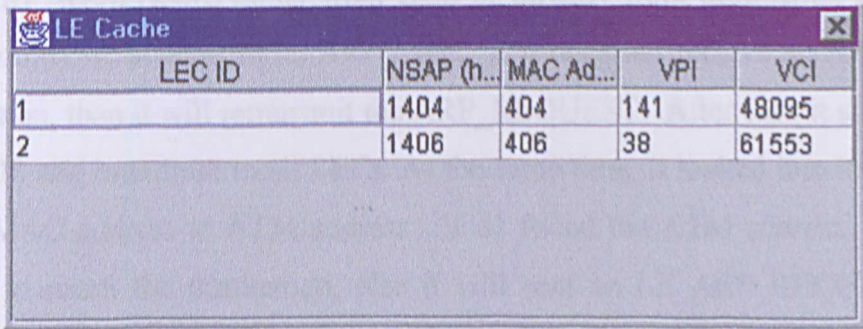
In the joining phase, all LECs that have the ATM address of LES will set up a VCC to contact the LES and sent LE_JOIN_REQUEST through the VCC. LECs that successfully join the ELAN will get a LECID from the its LES.

Expected Result

In this phase, c4 will not able to join comp ELAN because its MAC address is same with c2. c2 is registered earlier than c3 and therefore LES of comp will not allowed duplicate MAC address to register with it. Other LECs will successfully join the intended ELAN.

Simulation Result

As shown in Figure 6.3, only c1 and c2 is allowed to join the ELAN whereas c3 is rejected by LES of comp. Besides that, d1, d2, and d3 has successfully joined default ELAN.



LEC ID	NSAP (h...	MAC Ad...	VPI	VCI
1	1404	404	141	48095
2	1406	406	38	61553

Figure 6.3: LE Cache of comp LES

6.3.2.3 Data Transfer Phase

All LECs that successfully joined the ELAN will start sending data to the destination. In this phase, the ability of LEC to coin an ARP_REQUEST to get the destination MAC address and reply an ARP_RESPONSE after receiving an ARP_REQUEST for its AMC address will be tested. Besides that, a LEC will send LE_ARP_REQUEST to get the ATM address of the destination. The testing of LECs ability to retransmit ARP_REQUEST after control time out and clear its LE_ARP table after aging time is carried out.

LES/BUS of comp ELAN and default ELAN is test to multicast ARP_REQUEST and ARP_RESPONSE received from LECs, response to LE_ARP_REQUEST received from LECs.

In this phase, we will look into details the data transmission of d1. Table 6.9 shows the d1 data transfer time and destination:

Time (second)	Destination IP
0	202.185.2.4
3	202.185.2.3
5	202.185.2.4
9	202.185.2.3

Table 6.9: Data Transfer of d1

Expected Result

Every time when d1 need to transmit data, it will go through the data transfer phase. First of all, d1 will look into its ARP table in order to find MAC address of destination. If the MAC address is not found, it sent an ARP_REQUEST to BUS and broadcast to every LECs. Then

it will get ARP_RESPONSE either from BUS or directly from destination LEC and stored the destination MAC address in its ARP table. If it does not receive an ARP_RESPONSE from destination, then it will retransmit the ARP_REQUEST. After that, it start sending data packet to BUS and broadcast to all LECs. At the same time, it looked into its LE_ARP table to map the MAC address to ATM address. If d1 found the ATM address, it set up a data direct VCC to reach the destination, else it will sent an LE_ARP_REQUEST to LES to resolve the ATM address of destination. After it gets the ATM address from LES, it will set up a data direct VCC to sent data to the destination. This procedure is repeated in very data transmission session. d1 will clear its cache in LE_ARP table after the aging time is expired.

Simulation Result

In the simulation, d3 was not able to received ARP_REQUEST from d1 because it hasn't joined default ELAN when d1 sent the ARP_REQUEST. d1 retransmit the ARP_REQUEST after the control time out expired. Then, it received the ARP_RESPONSE through the Data Direct VCC from d3. After d1 received the ARP_RESPONSE from d3, it save the MAC address in to its ARP table and looked into its LE_ARP table to find for d3 ATM address but it is not available. So, it sent a LE_ARP_REQUEST to LES and LES sent d1 a LE_ARP_RESPONSE that attached the ATM address of d3. d1 save the ATM address of d3 into LE_ARP table and set up a new data direct VCC to send data. Then, it starts sending data and release the data direct VCC after the data transmission is over.

During the second data transfer session, it will go through the data transfer phase again but it received the ARP_RESPONSE from d2 through BUS. It release the data direct VCC after the data transmission is over.

In the third data transfer session to d3, it is able to find the MAC address and ATM address of d3. Then, it directly set up a new data direct VCC to contact the destination and start sending data to d3.

After aging time expired, d1 clear its LE_ARP table cache. So, it will need to go through the data transfer phase all over again to get the MAC address and ATM address of d2.

6.4 Summary

Testing of LANE stimulator begins with component testing, followed by module testing and system testing. Component testing focuses on the individual testing of each class. Meanwhile, module testing focuses on configuration phase testing, registration phase testing and data transfer phase testing. The system testing tests the whole simulator to ensure that it runs on the actual environment.

CHAPTER 7: Conclusion

The development of the ATM LANE network interface has provided useful and valuable insights into the technology of network interfaces, as well as into the viability of several important concepts used in the approach to network architecture.

CHAPTER 7

CONCLUSION

The strength of the Java programming language has enabled the creation of a powerful and elegant programming approach. It is this approach that is reflected in the design of the system and satisfies these principles. Java provides all the tools needed to create a class hierarchy, encapsulation, abstraction and polymorphism.

The object-oriented design of the system has enabled the system to be designed in a way that is easy to understand and use. The system is designed to be easy to use and to be easy to understand.

The system is designed to be easy to use and to be easy to understand. The system is designed to be easy to use and to be easy to understand. The system is designed to be easy to use and to be easy to understand.

7.1 System Strengths

- The design of network interface is simple, easy to use. The user can easily add a new LANE component to the existing and loading the network.
- The user can be fully aware of the status of the functions and modules are built in the system.
- The user can use the IPC to transmit data packets to desired destination by changing the destination address.

CHAPTER 7: Conclusion

The development of the ATM LANE network simulator components has provided several valuable insights into the methodology of network simulations, as well as proven the viability of several important concepts used in the approach to network simulation.

The use of Object-Oriented Programming approach has provided a several key benefits to the development of the network simulator. It provides the benefits of modularity in which every object forms a separate entity whose internal workings are decoupled from other parts of the system. The use of this approach provide the network simulator with other features such as simplicity, modularity, extensibility, maintainability and reusability.

The strength of the Java programming language has realized the benefits Object-Oriented Programming approach. It provides the capability to implement object-oriented principles and enforces these principles. Java provides all the luxuries of object-oriented programming: class hierarchy, inheritance, encapsulation and polymorphism.

The other great features of Java are built-in support for multithreading and the ability of the threads to run simultaneously. Finally, the ability of Java to work under different platforms has fulfilled the objectives of this project.

This project managed to achieve the overall project objectives and goals, i.e. development of an object-oriented, multithreading and cross-platform ATM LANE simulator. This simulator can be used as a training kit for user in designing and testing LANE. The following highlights the strengths, limitation and future enhancement of the LANE simulator.

7.1 System Strengths

- The design of network simulator is user friendly and easy to use. The user can easily add a new LANE component to the topology and simulate the network.
- The simulator is fully object-oriented whereby all the functions and modules are built in class.
- The user can use the LEC to transmit data packet to desired destination continuously by just changing the destination IP address.

7.2 System Limitation

- Location of LECS via ILMI is not supported.
- The Virtual Machine of Java cause the simulation runs slowly.
- The data packet does not go through ATM Adaptation Layer 5 (AAL 5), which perform the function of segmentation and reassembly the data packet.

7.3 Future Enhancement

- Supports for unregistration of LEC from its current ELAN. Currently, the LEC can register with an ELAN but it is not allowed to unregister itself from the ELAN.
- Implementation of Distributed LANE in order to distribute the LANE services load among a mesh of LES/BUS DLE peer servers. This will be able to improve the performance for remote LECs, increase the network reliability and fault tolerance.

[5] James V. Lexell et. Al. "NBIIA Next Hop Resolution Protocol (NHRP)", Inter Request For Comment 2372, Internet Engineering Task Force, April 1998

[6] LAN Emulation Sub-Working Group. "LAN Emulation over ATM Specification version 1.0", AT-LANE-0021.00, ATM Forum, January 1993.

[7] Multicast Sub-working Group. "Multicast Over ATM Specification version 1.0", AT-MPOA-0067.00, ATM Forum, July 1997.

[8] D. Ginsburg. "ATM Solutions for Enterprise Interconnecting", Addison Wesley, 1996.

[9] P.A. Henrich. "Computer Simulation: The Art and Science of Digital Worlds Construction", IEEE Potential, pp. 24-27, February/March 1996.

[10] Bruce A. Mac. INRAME 1.0a0, University of Berkeley, California, 1995/1996.

[11] N. Gidrai, P. Mouroux, L. Basso, Y. Samelhan, A. Krasny, D. Su. "The NIST ATM/ATM Network Simulator Operation and Programming Guide, Version 4.0", NISTIR 5703R2, National Institute of Standards and Technology, October 1998.

REFERENCES

- [1] Walter J. Goralski. "Introduction to ATM networking", McGraw-Hill, Inc., 1995.
- [2] William Stallings & Richard Van Slyke. "Business Data Communication", 3rd Edition, Prentice Hall Publishing, 1998.
- [3] Jun Xu. "IP over ATM: Classical IP, NHRP, LANE, MPOA, PAR and I-PNNI",
- [4] M. Laubach, J. Halpern, "Classical IP and ARP over ATM", Internet Request for Comment 2225, Internet Engineering Task Force, April 1998.
- [5] James V. Luciani et. Al. "NBMA Next Hop Resolution Protocol (NHRP)", Inter Request For Comment 2332, Internet Engineering Task Force, April 1998.
- [6] LAN Emulation Sub Working Group. "LAN Emulation over ATM Specification version 1.0", AF-LANE-0021.00, ATM Forum, January 1995.
- [7] Multiprotocol Sub-working Group. "Multi-protocol Over ATM Specification version 1.0", AF-MPOA-0087.000, ATM Forum, July 1997.
- [8] D. Ginsburg. "ATM: Solutions for Enterprise Internetworking", Addison Wesley, 1996.
- [9] P.A. Fishwick. "Computer Simulation: The Art and Science of Digital Worlds Construction", IEEE Potential, pp. 24-27. February/March 1996.
- [10] Bruce A. Mah. INSANE 1.0all, University of Berkeley, California, 1995/1996.
- [11] N. Golmie, F. Mouveaux, L. Hester, Y. Saintillan, A. Koenig, D. Su. "The NIST ATM/HFC Network Simulator Operation and Programming Guide Version 4.0", NISTIR 5703R2, National Institute of Standards and Technology, December 1998.

- [12] www.ifn.et.tu-dresden.de/TK/yats/yats.html
- [13] A. Varga et. Al. *OMNeT++ 1.1*, Technical University of Budapest, 1995.
- [14] Alex Maranda & Razvan Ghilea. "Inside NetSim++", 1996.

APPENDIX

A. System Testing Log File

ID 1 "lecs" "Status of LECS"
ID 2 "default" "Status of LES"
ID 4 "comp" "Status of LES"
ID 3 "d1" "Status of LEC"
ID 9 "d2" "Status of LEC"
ID 8 "d3" "Status of LEC"
ID 7 "c1" "Status of LEC"
ID 6 "c2" "Status of LEC"
ID 5 "c4" "Status of LEC"
0 3 find LECS
0 3 UNI to LECS
0 9 find LECS
0 9 UNI to LECS
0 8 find LECS
0 8 UNI to LECS
0 7 find LECS
0 7 UNI to LECS
0 6 find LECS
0 6 UNI to LECS
4418 3 Connection to LECS successfully
4418 3 Sending LE_CONF_REQUEST to LECS
5237 1 Get LE_CONF_REQUEST from1306
5237 1 Send success message to 1306
5510 7 Connection to LECS successfully
5510 7 Sending LE_CONF_REQUEST to LECS
5783 9 Connection to LECS successfully
5783 9 Sending LE_CONF_REQUEST to LECS
6056 3 Found LES from LECS
6056 6 Connection to LECS successfully
6056 6 Sending LE_CONF_REQUEST to LECS
6056 3 UNI to LES
6602 1 Get LE_CONF_REQUEST from1308
6602 1 Send success message to 1308
7148 3 UNI SUCCESS to LES
7148 1 Get LE_CONF_REQUEST from1404
7148 1 Send success message to 1404
7421 9 Found LES from LECS
7421 9 UNI to LES
7694 1 Get LE_CONF_REQUEST from1406
7694 1 Send success message to 1406
7967 3 Connection to LES successfully
7967 3 Sending LE_JOIN_REQUEST to LES
8513 9 UNI SUCCESS to LES
8786 7 Found LES from LECS
8786 2 Get LE_JOIN_REQUEST from 1306
8786 2 Response status 0 for 1306
8786 7 UNI to LES
9332 6 Found LES from LECS
9332 6 UNI to LES

9605 3 Join LES Successfully
9605 9 Connection to LES successfully
9605 3 Searching mac in ARP table failed
9605 3 Sending ARP Request
9605 9 Sending LE_JOIN_REQUEST to LES
9878 7 UNI SUCCESS to LES
10424 2 Get ARP REQUEST: from 202.185.2.2 finding 202.185.2.4
10424 2 Start Multicast ARP request
10424 6 UNI SUCCESS to LES
10697 7 Connection to LES successfully
10697 2 Get LE_JOIN_REQUEST from 1308
10697 2 Response status 0 for 1308
10697 7 Sending LE_JOIN_REQUEST to LES
11243 6 Connection to LES successfully
11243 1 Get LE_CONF_REQUEST from 1309
11243 1 Send Failure message to 1309
11243 6 Sending LE_JOIN_REQUEST to LES
11516 9 Join LES Successfully
11516 4 Get LE_JOIN_REQUEST from 1404
11516 4 Response status 0 for 1404
11516 9 Searching mac in ARP table failed
11516 9 Sending ARP Request
11789 8 Connection to LECS successfully
11789 8 Sending LE_CONF_REQUEST to LECS
12062 4 Get LE_JOIN_REQUEST from 1406
12062 4 Response status 0 for 1406
12335 2 Get ARP REQUEST: from 202.185.2.3 finding 202.185.2.2
12335 2 Start Multicast ARP request
12335 7 Join LES Successfully
12335 7 Searching mac in ARP table failed
12335 7 Sending ARP Request
12881 6 Join LES Successfully
12881 6 Searching mac in ARP table failed
12881 6 Sending ARP Request
13154 4 Get ARP REQUEST: from 202.185.3.2 finding 202.185.3.3
13154 4 Start Multicast ARP request
13427 1 Get LE_CONF_REQUEST from 1409
13427 1 Send success message to 1409
13700 4 Get ARP REQUEST: from 202.185.3.3 finding 202.185.3.2
13700 4 Start Multicast ARP request
15065 8 Found LES from LECS
15065 8 UNI to LES
16157 8 UNI SUCCESS to LES
17249 3 Get ARP request from 308 202.185.2.3
17249 3 Sent ARP RESPONSE to BUS
18068 2 Get ARP RESPONSE: from 202.185.2.2 to 202.185.2.3
18068 2 Sending Multicast ARP response
18341 6 Get ARP request from 404 202.185.3.2
18341 6 Sent ARP RESPONSE to BUS
18614 8 Connection to LES successfully
18614 7 Get ARP request from 406 202.185.3.3
18614 7 Sent ARP RESPONSE to BUS
18614 8 Sending LE_JOIN_REQUEST to LES
19160 4 Get ARP RESPONSE: from 202.185.3.3 to 202.185.3.2

19160 4 Sending Multicast ARP response
19433 4 Get ARP RESPONSE: from 202.185.3.2 to 202.185.3.3
19433 4 Sending Multicast ARP response
20252 2 Get LE_JOIN_REQUEST from 1409
20252 2 Response status 0 for 1409
21890 8 Join LES Successfully
21890 8 Searching mac in ARP table failed ...
21890 8 Sending ARP Request
23255 9 Get ARP response from 306
23255 9 Sending cell to LES for multicast
23528 2 Get ARP REQUEST: from 202.185.2.4 finding 202.185.2.2
23528 2 Start Multicast ARP request
24074 2 LE ARP request: get from 1308 finding 306
24074 2 Find ATM add
24074 7 Get ARP response from 406
24074 7 Sending cell to LES for multicast
24347 2 Get ARP Broadcast: from 308 sending to 306
24347 2 Multicast data cell
24893 9 Found atm addr from LES
24893 4 LE ARP request: get from 1404 finding 406
24893 4 Find ATM add
24893 6 Get ARP response from 404
24893 6 Sending cell to LES for multicast
25166 4 Get ARP Broadcast: from 404 sending to 406
25166 4 Multicast data cell
25712 4 LE ARP request: get from 1406 finding 404
25712 4 Find ATM add
25712 7 Found atm addr from LES
25985 4 Get ARP Broadcast: from 406 sending to 404
25985 4 Multicast data cell
26194 7 Sending cell to LES for multicast
26194 7 Sending setup cell 1406
26531 6 Found atm addr from LES
27077 4 Get ARP Broadcast: from 404 sending to 406
27077 4 Multicast data cell
27350 7 UNI SUCCESS to lec
27495 9 Sending cell to LES for multicast
27495 9 Sending setup cell 1306
27719 6 Sending cell to LES for multicast
27719 6 Sending setup cell 1404
28169 2 Get ARP Broadcast: from 308 sending to 306
28169 2 Multicast data cell
28169 7 Connection to LEC successfully
28169 7 Start sending cell
28442 9 UNI SUCCESS to lec
28442 4 Get ARP Broadcast: from 406 sending to 404
28442 4 Multicast data cell
28715 6 UNI SUCCESS to lec
28715 3 Get ARP request from 409 202.185.2.4
28715 3 Sent ARP RESPONSE to BUS
29261 9 Connection to LEC successfully
29261 9 Start sending cell
29534 2 Get ARP RESPONSE: from 202.185.2.2 to 202.185.2.4
29534 2 Sending Multicast ARP response

29534 6 Connection to LEC successfully
29534 6 Start sending cell
35813 8 Get ARP response from 306
35813 8 Sending cell to LES for multicast
37451 2 LE ARP request: get from 1409 finding 306
37451 2 Find ATM add
37724 2 Get ARP Broadcast: from 409 sending to 306
37724 2 Multicast data cell
39089 8 Found atm addr from LES
44293 8 Sending cell to LES for multicast
44293 8 Sending setup cell 1306
45368 8 UNI SUCCESS to lec
45914 2 Get ARP Broadcast: from 409 sending to 306
45914 2 Multicast data cell
47825 8 Connection to LEC successfully
47825 8 Start sending cell
500000 5 find LECS
500000 5 UNI to LECS
511652 5 Connection to LECS successfully
511652 5 Sending LE_CONF_REQUEST to LECS
513290 1 Get LE_CONF_REQUEST from 1408
513290 1 Send success message to 1408
514928 5 Found LES from LECS
514928 5 UNI to LES
516020 5 UNI SUCCESS to LES
516839 5 Connection to LES successfully
516839 5 Sending LE_JOIN_REQUEST to LES
517658 4 Get LE_JOIN_REQUEST from 1408
517658 4 Response status 5 for 1408
518477 5 Failed Joining LES
100009605 3 Can't get ARP RESPONSE from destination
100010324 2 Get ARP REQUEST: from 202.185.2.2 finding 202.185.2.4
100010324 2 Start Multicast ARP request
100016603 8 Get ARP request from 306 202.185.2.2
100016603 8 Sent ARP RESPONSE through conn
100018241 3 Get ARP response from 409
100018241 3 Sending cell to LES for multicast
100019060 2 LE ARP request: get from 1306 finding 409
100019060 2 Find ATM add
100019333 2 Get ARP Broadcast: from 306 sending to 409
100019333 2 Multicast data cell
100019879 3 Found atm addr from LES
100021067 3 Sending cell to LES for multicast
100021067 3 Sending setup cell 1409
100021790 2 Get ARP Broadcast: from 306 sending to 409
100021790 2 Multicast data cell
100022063 3 UNI SUCCESS to lec
100022354 7 Finish sending data cell
100023893 3 Sending cell to LES for multicast
100024520 3 Connection to LEC successfully
100024520 3 Start sending cell
100024793 2 Get ARP Broadcast: from 306 sending to 409
100024793 2 Multicast data cell
133324487 6 Finish sending data cell

150021735 9 Finish sending data cell
199993643 3 Finish sending data cell
299993643 3 Searching mac in ARP table failed
299993643 3 Sending ARP Request
299994290 2 Get ARP REQUEST: from 202.185.2.2 finding 202.185.2.3
299994290 2 Start Multicast ARP request
299999477 9 Get ARP resquest from 306 202.185.2.2
299999477 9 Sent ARP RESPONSE to BUS
300000296 2 Get ARP RESPONSE: from 202.185.2.3 to 202.185.2.2
300000296 2 Sending Multicast ARP response
300005210 3 Get ARP response from 308
300005210 3 Sending cell to LES for multicast
300006029 2 LE ARP request: get from 1306 finding 308
300006029 2 Find ATM add
300006302 2 Get ARP Broadcast: from 306 sending to 308
300006302 2 Multicast data cell
300006848 3 Found atm addr from LES
300008036 3 Sending cell to LES for multicast
300008036 3 Sending setup cell 1308
300008759 2 Get ARP Broadcast: from 306 sending to 308
300008759 2 Multicast data cell
300009032 3 UNI SUCCESS to lec
300009851 3 Connection to LEC successfully
300009851 3 Start sending cell
300032773 8 Finish sending data cell
399980612 3 Finish sending data cell
499980612 3 Succesfully get mac 409 from arp table
499980612 3 Find atm addr in LE_ARP table
499980612 3 Sending cell to LES for multicast
499980612 3 Sending setup cell 1409
499981259 2 Get ARP Broadcast: from 306 sending to 409
499981259 2 Multicast data cell
499981532 3 UNI SUCCESS to lec
499983438 3 Sending cell to LES for multicast
499983989 3 Connection to LEC successfully
499983989 3 Start sending cell
499984262 2 Get ARP Broadcast: from 306 sending to 409
499984262 2 Multicast data cell
500028169 7 Delete cache from LE_ARP table
500029261 9 Delete cache from LE_ARP table
500029534 6 Delete cache from LE_ARP table
500047825 8 Delete cache from LE_ARP table
599956014 3 Finish sending data cell
600024520 3 Delete cache from LE_ARP table
800009851 3 Delete cache from LE_ARP table
899956014 3 Succesfully get mac 308 from arp table
899956014 3 Can't find atm addr in LE_ARP table...sending LE_ARP request
899956014 3 Sending LE_ARP_REQUEST to LES
899956014 3 Sending cell to LES for multicast
899956835 2 LE ARP request: get from 1306 finding 308
899956835 2 Find ATM add
899957108 2 Get ARP Broadcast: from 306 sending to 308
899957108 2 Multicast data cell
899957654 3 Found atm addr from LES

899958840 3 Sending cell to LES for multicast
899958840 3 Sending setup cell 1308
899959565 2 Get ARP Broadcast: from 306 sending to 308
899959565 2 Multicast data cell
899959838 3 UNI SUCCESS to lec
899960657 3 Connection to LEC successfully
899960657 3 Start sending cell
999931416 3 Finish sending data cell
999983989 3 Delete cache from LE_ARP table

University of Malaya